THE UNIVERSITY *of* EDINBURGH

Unification-based Constraints for Statistical Machine Translation

Philip Williams

Doctor of Philosophy

Institute for Language, Cognition and Computation
School of Informatics
University of Edinburgh

2014

Abstract

Morphology and syntax have both received attention in statistical machine translation research, but they are usually treated independently and the historical emphasis on translation into English has meant that many morphosyntactic issues remain under-researched. Languages with richer morphologies pose additional problems and conventional approaches tend to perform poorly when either source or target language has rich morphology.

In both computational and theoretical linguistics, feature structures together with the associated operation of unification have proven a powerful tool for modelling many morphosyntactic aspects of natural language. In this thesis, we propose a framework that extends a state-of-the-art syntax-based model with a feature structure lexicon and unification-based constraints on the target-side of the synchronous grammar. Whilst our framework is language-independent, we focus on problems in the translation of English to German, a language pair that has a high degree of syntactic reordering and rich target-side morphology.

We first apply our approach to modelling agreement and case government phenomena. We use the lexicon to link surface form words with grammatical feature values, such as case, gender, and number, and we use constraints to enforce feature value identity for the words in agreement and government relations. We demonstrate improvements in translation quality of up to 0.5 BLEU over a strong baseline model.

We then examine verbal complex production, another aspect of translation that requires the coordination of linguistic features over multiple words, often with long-range discontinuities. We develop a feature structure representation of verbal complex types, using constraint failure as an indicator of translation error and use this to automatically identify and quantify errors that occur in our baseline system. A manual analysis and classification of errors informs an extended version of the model that incorporates information derived from a parse of the source. We identify clause spans and use model features to encourage the generation of complete verbal complex types. We are able to improve accuracy as measured using precision and recall against values extracted from the reference test sets.

Our framework allows for the incorporation of rich linguistic information and we present sketches of further applications that could be explored in future work.

Lay Summary

The field of machine translation was dramatically altered by IBM's introduction of statistical translation models in the early 1990s. Whereas previous approaches had required the painstaking manual development of translation dictionaries and grammars, IBM's models were able to learn statistical patterns of translation from previously-translated corpora and then translate new texts by generating and searching through huge numbers of potential translations. In the following years, the field of statistical machine translation (SMT) has made rapid progress.

Linguistically, the modelling of language in SMT is extremely simplistic — largely by necessity. A significant proportion of SMT research is dedicated to ameliorating modelling deficiencies, although doing so is rarely straightforward. Morphology and syntax have both received a considerable amount of attention and these efforts have been met with some success. However, these two aspects of grammar are usually treated independently, which precludes the accurate modelling of many linguistic phenomena. The historical focus on translation into English has meant that many morphosyntactic issues remain under-researched.

In both computational and theoretical linguistics, feature structures, which are simple, structured collections of attributes, and the associated information-combining operation of unification have proven powerful tools for modelling many aspects of language. In this thesis, we propose a framework that extends a state-of-the-art syntax-based SMT model by using feature structures to store morphosyntactic attributes of target-language words. We then apply constraints to ensure that syntactically-related words of the translation have compatible properties. Whilst our framework is language-independent, we focus on problems in the translation of English to German, a language pair that has a high degree of syntactic reordering and rich target-side morphology.

We first apply our approach to modelling agreement and case government phenomena. We use feature structures to associate target-side words with grammatical attributes, such as case, gender, and number, and we use constraints to ensure attribute compatibility for the words in agreement and government relations. We demonstrate improvements in translation quality over a strong baseline model.

We then apply our approach to modelling verbal complexes — multi-verb constructions where a main verb is used with auxiliaries (in English, examples are "will play," "has been playing," and "is played"). We find that using target-side constraints alone is insufficient for this task and we improve our model by incorporating additional information about the corresponding source-side clauses and their verbs.

Acknowledgements

I would first like to thank my supervisor Philipp Koehn. It was Philipp's teaching at the University of Edinburgh that introduced me to statistical machine translation and inspired me to begin research in the field. It was his support and guidance over the following five years that enabled me to complete this thesis.

I am greatly indebted to my thesis examiners, Bonnie Webber, Ondřej Bojar, and Marcello Federico, whose thoroughness and insight resulted in a much improved final draft of this thesis. Similarly, the feedback of Sharon Goldwater, Miles Osborne, and Barry Haddow during earlier stages of the PhD was invaluable.

The University of Edinburgh's School of Informatics is a wonderful environment to undertake research in natural language processing. I have learned a great deal from the school's researchers, particularly those of the SMT group, past and present: Abhishek Arun, Michael Auli, Alexandra Birch, Christian Buck, Loïc Dugast, Nadir Durrani, Yang Gao, Ulrich Germann, Liane Guillou, Barry Haddow, Eva Hasler, Kenneth Heafield, Hieu Hoang, Matthias Huck, Philipp Koehn, Abby Levenberg, Adam Lopez, David Matthews, Maria Nădejde, Miles Osborne, Hervé Saint-Amand, Rico Sennrich, and Bonnie Webber.

Research in statistical machine translation is predominantly experimental in nature and this work is no exception. The experiments undertaken in this thesis would not have been possible without the software provided by the Moses community. The Herculean coding efforts of Hieu Hoang especially deserve to be singled out for recognition.

The final person to thank is Gillian Foy. She assiduously proofread this thesis, but more importantly she provided ceaseless support and encouragement throughout its development and she made these last few years infinitely more enjoyable.

Lastly, I would like to acknowledge the sources of financial assistance that I received during my PhD. This work was primarily funded by the EPSRC. In addition, part of the research leading to these results received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement 287658 (EU-BRIDGE).

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Philip Williams*)

Contents

Chapter 1

Introduction

> *But when we say that a translation is an acceptable one, what we name is an overall relationship between source and target that is neither identity, nor equivalence, nor analogy — just that complex thing called a good match.* — David Bellos (2011)

The task of statistical machine translation is framed in the following way: given a string s in the source language, find the string t^* in the target language that has the highest probability according to the distribution $p(t|s)$. Of course, the true distribution $p(t|s)$ is unknowable without first solving virtually every problem in artificial intelligence, linguistics, and probably a number of fields yet to be invented. So instead we try to define a model that assigns higher probabilities to "good" translations and lower probabilities to "bad" translations.

Since the breakthrough development of IBM's word-based models in the late 1980s (Brown et al., 1990; 1993), the field has made rapid progress. Phrase-based models (Och, 2002; Marcu and Wong, 2002; Zens et al., 2002; Koehn et al., 2003) translate and reorder text in chunks, allowing the model to capture localized phenomena, such as the reordering of adjectives and nouns between French and English, the insertion or deletion of punctuation, and translations of multiword expressions and idioms like "Arthur's Seat" or "cry wolf." Syntax-based models have followed, with the emphasis being on the use of phrase-structure (Yamada and Knight, 2001; Galley et al., 2004; Chiang, 2005) or dependency structure (Shen et al., 2008) to improve word order.

However, most research on statistical modelling over this time has been focused on translation into English. Languages with richer inflectional morphologies pose additional challenges for translation and conventional SMT approaches tend to perform poorly when either source or target language has rich morphology (Koehn, 2005).

In this thesis our focus is on translation into morphologically-rich languages with

the aim of improving linguistic consistency of output. Previous approaches have successfully applied sequence models (Koehn and Hoang, 2007; Toutanova et al., 2008; Green and DeNero, 2012) to encourage consistent inflectional choices over adjacent words. This works well for many localised phenomena, but the models do not account for longer-range phenomena such as pronoun-antecedent agreement or subject-verb agreement in verb-final languages. To do this requires keeping track of both morphological and syntactic features simultaneously.

In both computational and theoretical linguistics, many monolingual models of phrase-structure grammar make use of feature structures to represent underlying linguistic properties of words and constituents. Feature structures together with the associated operation of unification have proven a powerful tool for modelling many aspects of natural language, enabling concise accounts of agreement, case control, verb subcategorization, and verb-raising, among others.

We believe that the well-defined machinery of feature structures and unification offer a means of incorporating further linguistic information into syntax-based models and tackling problems that are poorly addressed by surface form and tree structure representations alone. We therefore propose a framework that extends a strong syntax-based model with a feature structure lexicon and unification-based constraints.

Whilst our framework is language-independent, we focus on problems in the translation of English to German, a language pair which has a high degree of syntactic reordering and rich target-side morphology. Specifically, we apply the approach to agreement, case government, and verbal-complex translation. We also present sketches of further applications that could be explored in future work.

1.1 Thesis Outline

Chapter 2 outlines the three major types of model used in statistical machine translation: word-based, phrase-based, and syntax-based. Almost all of the related work that we subsequently refer to is rooted in one of these approaches. The emphasis in this chapter is on the models' generative stories and mathematical formulation.

Chapter 3 describes the main algorithms that have been developed for rule extraction and decoding in string-to-tree models, the type of syntax-based model that we use as our baseline. We will later adapt these standard rule extraction and decoding algorithms to incorporate unification-based constraints in the grammar and to enforce constraints during translation.

Chapter 4 introduces the fundamental concepts of unification-based approaches to grammar. We discuss the use of unification-based models in transfer-based approaches to machine translation.

Chapter 5 presents the unification-based framework that we use throughout the rest of the thesis. We first describe the grammar formalism, an extension of synchoronous context-free grammar that adds constraints to the target-side of the grammar rules. We then describe how constraint evaluation can be integrated into decoding.

Chapter 6 describes the baseline model and data that are used for experiments in subsequent chapters.

Chapter 7 discusses the problems posed by inflectional morphology in statistical translation and particularly issues of agreement and government. We show experimentally that naively integrating morphological tags can harm translation quality and we discuss previous approaches to the problem. We apply our unification-based approach to specific problems of agreement and case government in German.

Chapter 8 examines verbal-complex production, another aspect of translation that requires the coordination of linguistic features over multiple, often discontiguous, words. The problem is again morphosyntactic, but with a greater intermixture of syntactic and morphological form than is seen in agreement or government. We discuss the multiple sources of model error that can contribute to translation failure before developing a unification-based model of verbal-complex production. We use the failure to form satisfactory feature structures as an indicator of translation error and use this to automatically identify and quantify errors that occur in our baseline system. A manual analysis and classification of these errors informs the next chapter.

Chapter 9 extends our verbal-complex model to incorporate source-side information into the constraints. We use syntactic information from the source to identify clause spans and use model features to encourage the generation of complete verbal-complex types. We measure accuracy against values extracted from the reference test sets.

Finally, Chapter 10 sketches other applications for our framework that could be explored in future work and concludes.

1.2 Thesis Contributions

- We develop a language-independent framework for incorporating additional linguistic information into syntax-based translation models using the well-understood

machinery of feature structures and unification.

- We describe how unification-based constraint evaluation can be efficiently integrated into parsing-based decoding. We demonstrate that, although there are computational costs, our approach is viable for full-scale translation tasks.

- We develop models for agreement, government, and verbal-complex translation in German, and demonstrate improvements in translation quality over a strong baseline model.

- We demonstrate that constraints can be useful for pinpointing translation errors in a system. For our model of verbal-complex production, we find that approximately 80% of incomplete and inconsistent feature structures indicate genuine translation errors. Using this as the basis of a semi-automatic approach to analysis, we present a fine-grained error classification for our baseline system.

- Finally, we outline further applications that could be addressed in future work.

1.3 Related Publications

Chapters 5 and 7 expand on work that was published under the title "Agreement Constraints for Statistical Machine Translation into German" in the Proceedings of the Sixth Workshop on Statistical Machine Translation (Williams and Koehn, 2011).

The baseline system described in Chapter 6 and used throughout the thesis is based on the system described in the paper "GHKM Rule Extraction and Scope-3 Parsing in Moses" published in the Proceedings of the Seventh Workshop on Statistical Machine Translation (2012) (Williams and Koehn, 2012).

Chapters 8 and 9 present an expanded version of work that was published under the title "Using Feature Structures to Improve Verb Translation in English-to-German Statistical MT" in the Proceedings of the 3rd Workshop on Hybrid Approaches to Machine Translation (Williams and Koehn, 2014).

Chapter 2

Statistical Machine Translation Models

> *But it must be recognized that the notion "probability of a sentence" is an entirely useless one, under any known interpretation of this term.* — Noam Chomsky (1969)

2.1 Introduction

In this chapter we outline the three major types of model used in statistical machine translation: word-based, phrase-based, and syntax-based. These three approaches decompose the full problem of translation in different ways, making different simplifying assumptions, but they all fit within a common statistical framework that has its roots in information theory and machine learning.

The task of statistical machine translation is framed in the following way: given a string, s, in the source language, find the string, t^*, in the target language that has the highest probability according to a distribution $p(t|s)$:

$$t^* = \arg\max_t p(t|s) \tag{2.1}$$

The challenge is threefold: first, to model a probability distribution $p(t|s)$ that, given suitable parameters, assigns relatively high probabilities to "good" translations of s and relatively low probabilities to "bad" translations. Second, to learn the parameters of the model. And third, to provide a practical means of finding, or approximating, the highest probability target string t^* among a potentially infinite number of candidates.

In the following sections, we outline the statistical models that are used to define $p(t|s)$. Our main emphasis is on the generative stories and linguistic expressiveness of the models. We defer discussion of how the models are trained and the computational search process to later chapters.

Almost all of the related research uses one of these three model types and this chapter introduces concepts that are referred to later in the thesis.

First, we take a step back and consider the task of translation in terms that are more familiar to the human translator. Ultimately, we are trying to mimic human translation — the observed results, if not the underlying processes (which we are far from understanding). So what are the kinds of process that we are hoping to capture in our models?

2.2 The Translation Task

To better illustrate the strengths and weaknesses of these different models, we first examine two examples of human translation and draw attention to aspects of translation that we would like an ideal statistical model to capture. We will use these translations as the basis for running examples.

2.2.1 Example 1

Our first example sentence pair is taken from the development set that we will describe in Chapter 6. The original sentence is from an English-language article in The Economist magazine and was translated into German by a human translator.

English	As British political scandals go, this one is not particularly juicy.
German	Für britische Skandale ist dieser nicht besonders schlüpfrig.
Gloss	for British scandals is this not particularly juicy.

Figure 2.1: Sentence 1,460 from the newstest2008 development set

Apart from the dropping of the adjective 'political' the translation is faithful to the original. The simplest statistical approach, a direct word-for-word translation based on a frequency dictionary, might produce a gloss that a human reader could interpret with moderate success, but clearly we would prefer a translation that is closer to the human translator's above. Let us consider some of the aspects of this translation that we would like a statistical system to reproduce:

Idiomatic constructions The English sentence uses the adverbial construction *as . . . go*, which cannot be translated literally into German. In this case, the translator chooses a prepositional phrase similar to the English *for . . . .*

Verb placement The position of German verbs is fixed according to the clause type. For a declarative main clause, the finite verb always appears in the second position. This requires a change in constituent order relative to the English sentence, where the finite verb occurs in the third position (after the adverbial clause and the subject).

Case marking The subject of the English sentence is *this one*. Its role as subject is apparent from the syntactic structure of the clause: it appears before the finite verb in a declarative clause. German uses a freer verb argument order than English and the subject role of the translation, *dieser*, is instead indicated through inflection (if *this one* were instead the object it would be translated as an accusative form, such as *diesen*).

Noun-modifier agreement Unlike in English, German attributive adjectives are inflected to agree with the nouns that they modify. Since the noun *Skandale* is plural and occurs in an accusative-case phrase, the modifying adjective *britische* must be used rather than one of the four other attributive forms: *britischen, britischem, britischer, britisches*.

2.2.2 Example 2

Our second example is taken from the same development set as the first. The original sentence is from an English-language article in The New York Times and was translated by a human translator.

English	Members of the general public could buy tickets for 30 euros ($44.57).
German	Die Öffentlichkeit konnte Tickets für 30 Euros (44.57 Dollar) kaufen.
Gloss	the public could tickets for 30 Euros (44.57 dollars) buy.

Figure 2.2: Sentence 1,573 from the newstest2008 development set

Paraphrasing The translator chooses to render the English *members of the general public* as *die Öffentlichkeit* ("the public") instead of the closer possibility *Mitgliedern der Öffentlichkeit*. The longer form has a very similar meaning in both languages, but is used with differing frequencies (in the Europarl corpus, 'members of the public' occurs 203 times while 'Mitglieder(n) der Öffentlichkeit occurs 10 times).

Verb-final constructions In many German constructions, the main verb is placed at the end of a clause. When translating an English sentence, this can involve the arbitrarily large movement of the main verb relative to its original position. The grammar rules that decide verb-final placement are very regular: in this instance, it is the use of a modal finite auxiliary construction: *konnte* ... *kaufen*.

Subject-verb agreement Both English and German require present-tense finite verbs to agree in person and number with the subject. In Example 1, the subject ('this one') is singular and in the third person and so the verb form 'is' is used, rather than 'am' or 'are.' For most English verbs, only two finite present-tense forms are distinguished (via the presence or absence of the suffix '-s'). Modal verbs are an exception and in this example the same form, 'could,' would be used if the subject were singular. The same is not true of German, which requires agreement of finite modals and distinguishes a richer set of forms. For instance, 'could' would we be translated 'konntest' in the second-person singular.

2.3 Word-Based Models

The development in the late 1980s of IBM's word-based models (Brown et al., 1990; 1993) was a breakthrough in machine translation. Whilst these models are no longer state-of-the-art, having been supplanted by phrase-based models in the early 2000s, many of the concepts introduced with this work are still present in some form in contemporary models, and word-based models themselves are still used for the sub-task of automatic word alignment and as the basis of translation scoring features.

2.3.1 Word Alignments

A fundamental concept in word-based models is that of word-by-word alignments. In its most general form, a word alignment is a function that defines a many-to-many relationship between the source and target words of a sentence pair. Figure 2.3 illustrates two possible alignments for our example translation from Section 2.2.1.

Whilst the definition permits alignment links between any pair of words, it is clear that among all possible links, some reflect an underlying translational equivalence better than others. For example, the first alignment in Figure 2.3 defines a mapping between 'nicht' and 'not' which is clearly a truer equivalence than the second alignment's mapping between 'nicht' and 'this one is not particularly juicy.'

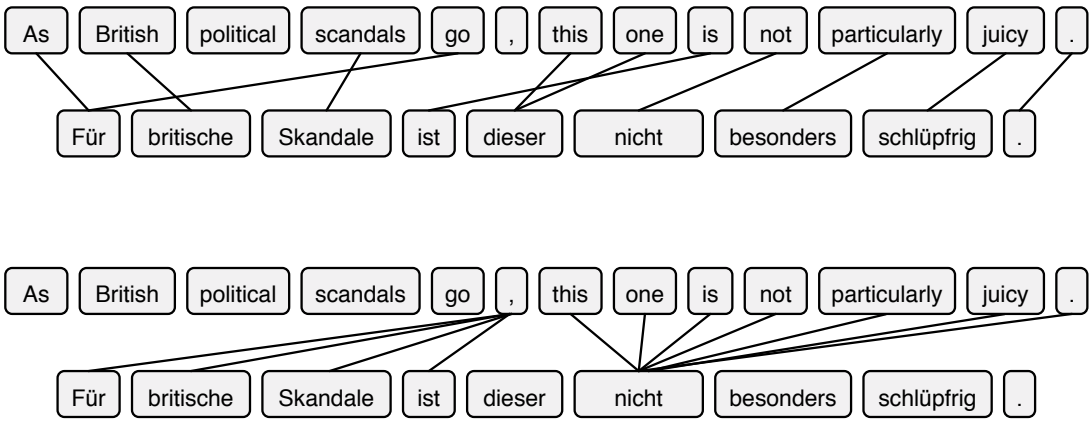


Figure 2.3: Two possible word alignments

2.3.2 The Noisy-Channel Model

Following an information theoretic approach that had recently been applied in automatic speech recognition, Brown et al. (1993) applied Bayes' theorem to obtain

$$p(t|s) = \frac{p(s|t)p(t)}{p(s)} \tag{2.2}$$

For any given source sentence, the denominator $p(s)$ is fixed and can therefore be factored out when comparing target string probabilities. The translation task is thus defined as the search for the target string t^* such that

$$\begin{aligned} t^* &= \arg\max_t p(t|s) \\ &= \arg\max_t p(s|t)p(t) \end{aligned} \tag{2.3}$$

In speech recognition, the equivalent $p(t)$ component had been successfully modelled using m-gram language models. Brown et al. (1993) adopted the same solution, making the $p(s|t)$ component their focus of attention.

2.3.3 IBM Models 1-5

As the term 'word-based' suggests, the fundamental unit used by the IBM models is the word. To model $p(s|t)$, the sentences s and t are treated as sequences of words, which we will denote as $s_1, s_2, \ldots, s_{|s|}$ and $t_1, t_2, \ldots, t_{|t|}$, respectively, and the problem of sentence translation is cast as a problem of combining lexical translations.

All five models presuppose the existence of many-to-one alignments from s to t, treating the alignment object as a hidden variable. The probability $p(s|t)$ can therefore be expressed as a sum over every possible many-to-one aligment a from s to t:

$$p(s|t) = \sum_a p(s,a|t) \tag{2.4}$$

As we have already noted, some alignments are more plausible than others and this is where the models differ: in Model 1 all alignments are assumed to have uniform probability. In the higher models, alignment weighting models become increasingly sophisticated.

Model 1

In common with the other four models, IBM Model 1 makes the simplifying assumption that the probability of a word s_i being produced from an aligned target word t_j depends only on the word t_j. In order to explain source words that have no natural target counterpart, a special target word, *null*, is added and 'unaligned' source words are aligned to *null*. Under the restriction that alignments are many-to-one, this gives a total of $(|t|+1)^{|s|}$ possible alignments. IBM model 1 is therefore expressed in the following equation:

$$p(s|t) = \frac{\varepsilon}{(|t|+1)^{|s|}} \sum_a \prod_{i=1}^{|s|} p(s_i|t_{a_i}) \tag{2.5}$$

where ε is a normalizing constant.

Model 2

Model 2 introduces the concept of a distortion function, d, a probability distribution where reordering of words is conditioned on $|s|$, $|t|$, and absolute target word position, i.

$$p(s|t) = \varepsilon \sum_a \prod_{i=1}^{|s|} p(s_i|t_{a_i}) d(a_i|i,|s|,|t|) \tag{2.6}$$

One possible definition of d is $1/(|t|+1)^{|s|}$ and therefore Model 1 is a special case of Model 2.

Models 3-5

Models 3-5 progressively introduce improvements over Model 2, at the cost of additional training complexity.

Model 3 refines Model 2 by introducing the notion of *fertility*. This corresponds to the intuitive idea that some words are more likely than others to be dropped during translation, whilst some words — like compounds or contractions — are more likely than others to be translated into multiple words. The fertility model is defined formally as a probability distribution $p(n|w)$ that indicates how likely it is that a word w will be be translated into n words.

Model 4 refines Model 3 by replacing the absolute distortion function with a relative model. Essentially, the relative distortion model is a probability distribution that indicates how likely it is that a translation of word t_{i+1} is placed at a distance d from the translation of word t_i.

Model 5 refines Model 4 by addressing a technical deficiency whereby probability mass is assigned to events that are not actually possible.

2.3.4 Shortcomings of Word-based Models

The suitability of a particular statistical approach to translation will depend to some extent on the characteristics of the source and target languages that are involved. For instance, making the word an atomic unit of translation has different implications for a minimally-inflected language like Chinese compared to a highly-inflected language like Finnish. All of the approaches in this chapter treat the surface-forms of words as atomic and we will return to this issue in Chapter 7 where we will describe some of the extensions and related models that have been proposed in the literature.

For translation between lightly-inflected languages, a more prominent shortcoming of word-based models is the assumption of independence between lexical translations. Many aspects of language and of translation are difficult to account for in terms of individual words and would be better explained by many-to-many translation rules. For instance, in the example translation of Section 2.2.1 the construction “as X go” was translated to “für X.” The generative story of IBM Model 4 goes something like this:

1. For the word ‘as’ choose a fertility of 0; for ‘go’ choose 1 (or vice versa)
2. Translate ‘go’ (or ‘as’) to ‘für’

3. Choose fertilities and translate the individual words of X

4. Reorder the resulting German words so that 'für' occurs before the translations of X's words (if we chose 'as' then no reordering is necessary)

The problem is that although translating "as X go" to "für X" may be quite likely, the individual steps of the above generative story are not: *für* is not a good translation of either *as* or *go*. Fertility and distortion scoring also has little context to go on.

2.4 Phrase-Based Models

Phrase-based models (Och, 2002; Marcu and Wong, 2002; Zens et al., 2002; Och and Ney, 2004) decompose sentence translation into a problem of combining phrasal translations. A 'phrase' in this case is simply a substring and is not related to the conventional linguistic notion of a phrase. The source and target sentences s and t are thus sequences of some number, $1 \leq L \leq min(|s|,|t|)$, of phrases. Phrasal alignment is simpler than word alignment: source and target phrases are aligned one-to-one and there is no `null` phrase. We will use the term *derivation* to refer jointly to a segmentation of a source sentence together with a one-to-one alignment to a sequence of target phrases. For a derivation that segments the source sentence into L phrases, we will denote the resulting sequence of target phrases as $\bar{t}_1, \bar{t}_2, \ldots, \bar{t}_L$ and we will denote the source phrase corresponding to $\bar{t}_i$ as $\bar{s}_i$ (the source sentence is thus a permutation of the sequence $\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_L$).

Figure 2.4 illustrates two possible derivations for the example translation of Section 2.2.1. The first involves comparatively large phrase pairs, which are able to account for localised phenomena such as the deletion of the comma and the translation of the phrasal construction *as ... go* to *für*. However, translation relies upon the model's repository of phrase-pairs and longer source phrases are likely to be sparse or unseen in the training data. The second derivation shows an alternative generative story that is more realistic under these conditions.

Early phrase-based models followed the noisy-channel approach used in word-based models. The objective is therefore as in Equation 2.3:

$$t^* = \arg\max_t p(s|t)p(t)$$

Just as the word-based models involve a sum over all possible alignments, the phrase-

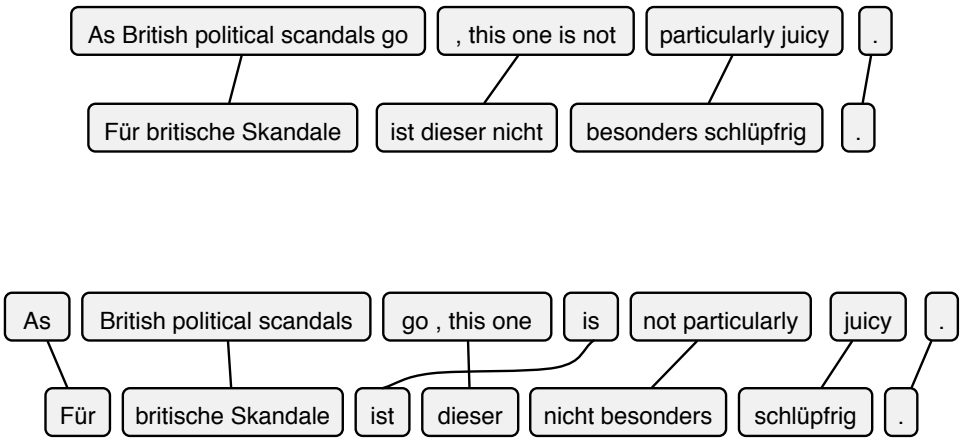


Figure 2.4: Two possible phrase-based derivations.

based model involves a sum over all possible derivations:

$$t^* = \arg\max_t \sum_d p(s,d|t)p(t)$$

where $d \in D(s,t)$, the set of derivations over s and t. To avoid the computationally intractable sum over derivations, the search objective is approximated using a single derivation:

$$t^* \approx \arg\max_{t,d} p(s,d|t)p(t) \tag{2.7}$$

As in the word-based models, $p(t)$ is modelled using an m-gram language model, leaving $p(s,d|t)$ as the focus of attention. Here is the formulation from Koehn et al. (2003):

$$p(s,d|t) = \prod_{i=1}^{L} p(\bar{s}_i|\bar{t}_i)r(a_i - b_{i-1}) \tag{2.8}$$

The function r is a probability distribution that models relative distortion. The argument is the difference between a_i and b_{i-1}, where a_i is the position of the first word of $\bar{s}_i$ and b_{i-1} is the position of the last word of $\bar{s}_{i-1}$. In other words, $a_i - b_{i-1}$ is a distance-based measure of the degree of source phrase reordering with respect to the aligned target phrases. In use, r is defined such that phrase reordering is penalized. Later work (Tillmann, 2004; Koehn et al., 2005) introduces more sophisticated reordering models that take into account the lexical content of the phrases.

2.4.1 Log-linear Models

At around the time the early phrase-based models were being developed, Och and Ney (2002) proposed a more general framework in place of the noisy-channel model.

Following the maximum entropy approach of Berger et al. (1996), they reformulated the fundamental equation of statistical machine translation as

$$p(t|s) = \frac{\exp\left(\sum_{m=1}^{M} \lambda_m h_m(t,s)\right)}{\sum_{t'} \exp\left(\sum_{m=1}^{M} \lambda_m h_m(t',s)\right)} \tag{2.9}$$

where $h_1, \ldots, h_M$ are real-valued functions and $\lambda_1, \ldots, \lambda_M$ are real-valued constants.

The denominator in Equation 2.9 is a sum over all possible translations of s. Computing its value is clearly a computationally demanding proposition for non-trivial functions h_m (and non-zero λs) even if the length of a translation is bounded. Fortunately, this term is constant for a given source sentence and so can be ignored for the purposes of decoding.[1] The noisy-channel formula of Equation 2.3 is thus replaced with

$$\begin{aligned} t^* &= \arg\max_t p(t|s) \\ &= \arg\max_t \sum_{m=1}^{M} \lambda_m h_m(t,s) \\ &= \arg\max_t \sum_d \sum_{m=1}^{M} \lambda_m h_m(t,s,d) \end{aligned} \tag{2.10}$$

As before, the search objective is approximated by a search for the best derivation:

$$d^* = \arg\max_d \sum_{m=1}^{M} \lambda_m h_m(t,s,d) \tag{2.11}$$

As Och and Ney (2002) point out, the noisy-channel formulation is a special case of Equation 2.11 (when $M = 2$, $\lambda_1 = \lambda_2 = 1$, $h_1 = \log p(s,d|t)$ and $h_2 = \log p(t)$). The log-linear formulation has the advantage that arbitrary feature functions can be added to the model and that the individual components can be weighted using the λ parameters. They demonstrated that this led to improvements in translation quality.

The log-linear approach was rapidly adopted and a common core of feature functions established. A modern phrase-based system typically includes feature functions for the language model, phrasal translation probabilities (in both directions), lexical translation probabilities (in both directions), a target-sentence length penalty, phrase count penalty, and a lexicalised reordering model.

[1] The denominator (or an approximation to it) may be required for training, depending on the approach. For instance, Och and Ney (2002) use the Generalized Iterative Scaling algorithm (Darroch and Ratcliff, 1972) to learn values for $\lambda_1, \ldots, \lambda_M$ that maximise the likelihood of the training data according to the model. To perform the renormalization required by the algorithm, they use a sampling-based approach to approximate the denominator. Starting with MERT (Och, 2003), most modern training approaches use task-specific criteria, such as the optimization of BLEU score, which do not require renormalization.

2.4.2 Shortcomings of Phrase-based Models

The phrase-based model as described in Section 2.4 defines a search problem that is exponential in sentence length (Knight, 1999). In practice, reordering must be reduced to a small window, which precludes many of the reorderings necessary in translating, for example, from an SVO language to an SOV language. Even for short and medium-range reordering, analysis has shown the phrase-based model to perform poorly (Birch et al., 2009).

The definition of translation rules as substring pairs means that many potentially useful generalizations cannot be learned. For example, knowing the mapping from 'as British political scandals go' to 'für britische Skandale' is of no help in translating 'as Polish political scandals go.'

In common with the word-based models, the phrase-based model has no means of incorporating linguistic structure, such as syntax, morphology, or semantics, beyond the surface forms.

2.5 Syntax-Based Models

2.5.1 Hierarchical Phrase-based Models

Hierarchical phrase-based models (Chiang, 2005; 2007) generalize the concept of a phrase to allow gaps into which other hierarchical phrases can be nested. Figure 2.5 illustrates one possible hierarchical phrasal segmentation for the example translation of Section 2.2.1.

Figure 2.5: A possible hierarchical phrasal alignment

The model is formalized as a synchronous context-free grammar (SCFG) (Aho and Ullman, 1969) in which gaps are represented by non-terminals with the generic label X. The grammar rules (with two exceptions) are of the form

$$X \rightarrow \langle \gamma, \alpha, \sim \rangle$$

where γ is a string of source terminals and non-terminals, α is a string of target terminals and non-terminals, and $\sim$ is a one-to-one correspondence between source and target non-terminals.

The segmentation shown in Figure 2.5 can be viewed as a sequence of three partial derivations (the first yields "as British political scandals go" on the source side, the second "this one is not particularly juicy", and the third "."). The first can be derived using the following SCFG rules (we use subscripted indices to denote the rules' non-terminal correspondences):

$$\begin{aligned} X &\rightarrow \langle \text{für } \mathrm{X}_{[1]}, \text{as } \mathrm{X}_{[1]} \text{ go} \rangle \\ X &\rightarrow \langle \text{britische Skandale}, \text{British political scandals} \rangle \end{aligned}$$

The grammar contains two further rules:

$$\begin{aligned} S &\rightarrow \langle \mathrm{X}_{[1]}, \mathrm{X}_{[1]} \rangle \\ S &\rightarrow \langle \mathrm{S}_{[1]}\ \mathrm{X}_{[2]}, \mathrm{S}_{[1]}\ \mathrm{X}_{[2]} \rangle \end{aligned}$$

S is the start symbol. Together, the two S rules, referred to as "glue" rules, can monotonically combine X derivations to produce S derivations. In Figure 2.5, the glue rules can be thought of as combining the sequence of partial derivations into a full sentence pair derivation.

Whereas the phrase-based model decomposed sentence translation into a problem of i) segmenting the source into phrases, ii) translating the phrases, and then iii) reordering the translations, the hierarchical model decomposes sentence translation into a process of i) segmenting the source sentence into hierarchical phrases and then ii) translating the hierarchical phrases. In the SCFG framework, the result is a synchronous derivation d, which is a sequence of rule applications.

As with modern phrase-based systems, Chiang (2005) formulated the hierarchical phrase-based model within the log-linear framework of Och and Ney (2002):

$$t^* = \arg\max_t \sum_d \sum_{m=1}^{M} \lambda_m h_m(t, s, d) \tag{2.12}$$

In this instance, $d \in D$, the set of synchronous derivations with source s and yield t. The set of feature functions is analagous to those of the phrase-based model, including hierarchical phrasal translation probabilities, lexical translation probabilities, word and rule count penalties, and an m-gram language model.

Again, the search is approximated by a search for a single derivation:

$$\begin{aligned} d^* &= \arg\max_d \sum_{m=1}^{M} \lambda_m h_m(d,s) \\ &= \arg\max_d \left(\lambda_1 \log p_{LM}(d) + \sum_{r_i} \sum_{m=2}^{M} \lambda_m h_m(r_i) \right) \end{aligned} \tag{2.13}$$

where r_i is a rule application in the derivation d. Whilst the derivation's score would ideally be decomposed fully into a sum of subderivation scores, the m-gram language model's cannot be decomposed in this way. We will return to this issue in Section 3.3 when we describe search algorithms for this model.

2.5.2 String-to-Tree Models

Chiang's (2005) model is syntactic in a purely formal sense. It is able to capture hierarchical structure inherent in the data, but it does not make use of linguistically-motivated syntactic annotation, which can be both a strength and a weakness.

If syntactic trees are available for the target-side of the training data then a richer model, often called a string-to-tree model[2], can be learned. Motivated by the potential for a stronger model of reordering, Yamada and Knight (2001) developed an early noisy-channel model in which a target parse tree is recovered from a source string that is presumed to have been transformed via a series of reordering, insertion, and translation operations. More recent work, such as that of Galley et al. (2004), Galley et al. (2006), Marcu et al. (2006), Zollmann and Venugopal (2006), and Zhang et al. (2011) is closer to the hierarchical phrase-based model.

Figure 2.6 shows part of the hierarchical phrasal segmentation from Figure 2.5 annotated on the English side with a phrase structure tree fragment. In this example, we treat the English side as the target-side, making this a string-to-tree model.

Two aspects are particularly important. The first is that the segmentation decomposes the target sentence into substrings that are each matched by a single subtree. The second is that the annotation is multi-level: that is, the annotation includes intermediate structure between the subtree root and the words and gaps (like the internal S and VP in the outer phrase's annotation in Figure 2.6). The use or not of intermediate structure determines the variety of synchronous grammar required to formalise the model.

[2]On this pattern, a hierarchical-phrase based model is sometimes called a string-to-string model, referring to the fact that the model operates on string pairs with the formal syntactic representation being somewhat latent.

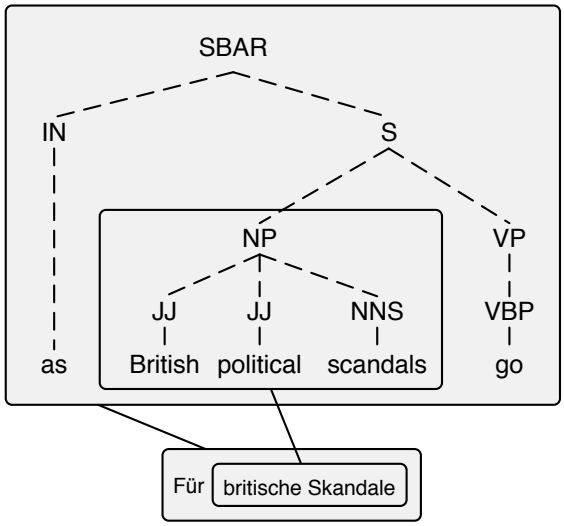


Figure 2.6: Hierarchical phrasal alignment with tree annotation

If intermediate structure is used then the model can be formalised as a synchronous tree-substitution grammar (STSG), a variant of synchronous tree-adjoining grammar (Shieber and Schabes, 1990) that includes the substitution operation but not the adjunction operation.

The STSG grammar rules are of the form

$$\langle \gamma, \pi, \sim \rangle$$

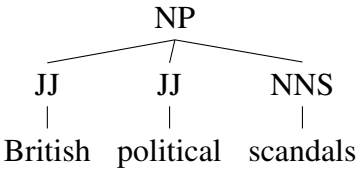
where γ is a string of source terminals and non-terminals, π is a tree with target terminal and non-terminal leaves, and $\sim$ is a one-to-one correspondence between source and target non-terminals.

Adopting the same convention of using subscripted indices for non-terminal correspondences, the STSG rules for Figure 2.6 can be written

$$\mathrm{X}_{[1]}\ , \qquad \begin{array}{c} \mathrm{S} \\ | \\ \mathrm{SBAR}_{[1]} \end{array} \tag{r1}$$

$$\text{für}\ \mathrm{X}_{[1]}\ , \qquad \begin{array}{ccc} & \mathrm{SBAR} & \\ \mathrm{IN} & \mathrm{NP}_{[1]} & \mathrm{VP} \\ | & & | \\ \text{as} & & \mathrm{VBP} \\ & & | \\ & & \text{go} \end{array} \tag{r2}$$

britische Skandale , (r3)



The internal structure of π does not influence rule application and therefore an STSG is always weakly-equivalent to an SCFG in which the πs are 'flattened' and their root labels converted to rule left-hand sides. In this case, the model can be formalised identically to the hierarchical phrase-based model except that the non-terminal vocabulary is expanded to include the tree labels and additional glue rules are included to combine derivations with label types from this richer vocabulary.

If we ignore the internal structure of Figure 2.6 then the corresponding derivation can be produced using the following SCFG rules:

$$\begin{aligned} S' &\rightarrow \langle \mathrm{X}_{[1]}, \mathrm{SBAR}_{[1]} \rangle \\ \mathrm{SBAR} &\rightarrow \langle \text{für } \mathrm{X}_{[1]}, \text{as } \mathrm{NP}_{[1]} \text{ go} \rangle \\ \mathrm{NP} &\rightarrow \langle \text{britische Skandale}, \text{British political scandals} \rangle \end{aligned}$$

We use S' as the start symbol avoid confusion with the constituent label S.

The string-to-tree model decomposes sentence translation into a process of i) segmenting the source sentence into hierarchical phrases and then ii) translating the hierarchical phrases, which generates a target tree. Apart from the constraint that the target tree is well-formed, the process is the same as for the hierarchical phrase-based model. The formulation of the model given in Equation 2.13 therefore carries over to the string-to-tree scenario unchanged, though of course the option opens up to define feature functions in terms of the syntactic annotation.

2.5.3 Tree-to-String Models

Tree-to-string models are formalized exactly as for string-to-tree models except that tree structure annotation is added to the source-side of the segmentation instead of the target-side. This alters the generative process, introducing the requirement that a parse tree is available for the source sentence. Thus, the tree-to-string model decomposes sentence translation into a process of i) segmenting the source tree into syntactically-annotated hierarchical phrases and then ii) translating the hierarchical phrases, which generates a target string.

Early tree-to-string models were developed by Huang et al. (2006) and Liu et al. (2006). The former refer to the approach as syntax-directed translation and as their

name suggests, the motivation for the incorporation of source syntax lies in the use of source language analysis as additional context for the selection of translation rules.

Tree-to-string systems have the practical advantage that the translation step is asymptotically faster than in hierarchical or string-to-tree models. Huang and Mi (2010) present a tree-to-string search algorithm that has linear time complexity with respect to sentence length.

2.5.4 Tree-to-Tree Models

Figure 2.7 shows a hierarchical phrasal segmentation annotated with phrase structure tree fragments on both the source and target sides.

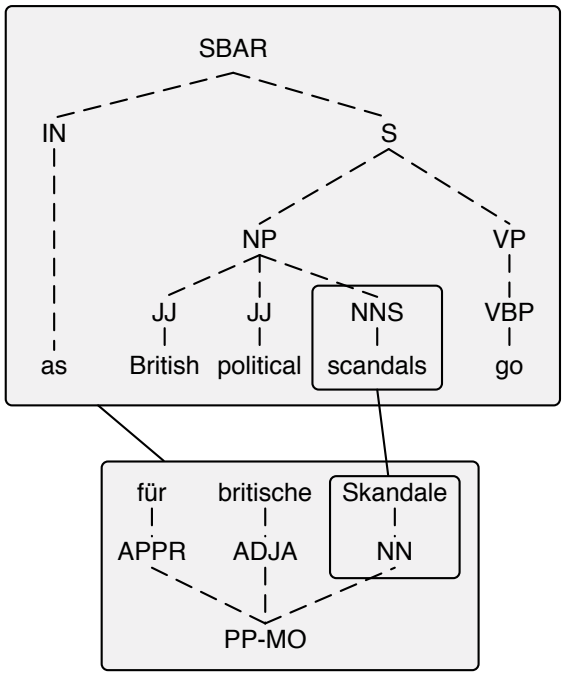


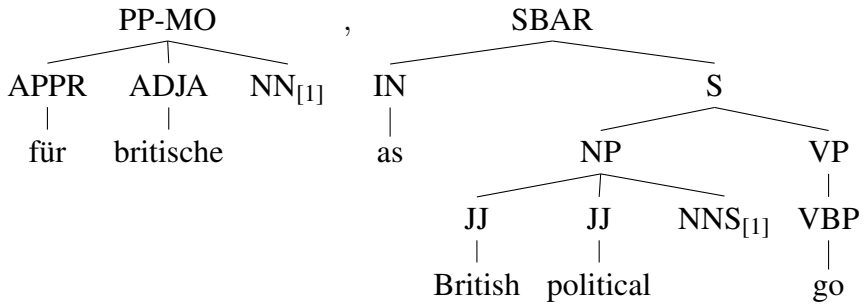
Figure 2.7: Hierarchical phrasal alignment with source and target tree annotations

Whereas the segmentation in Figures 2.5 and 2.6 contained a mapping between the phrases ‘britische Skandale’ and ‘British political scandals,’ such a segmentation is not possible here due to the non-isomporphic nature of the phrase-structure trees (there is not a tree node covering the two words ‘britische Skandale’).

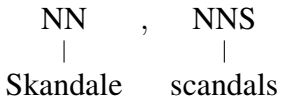
The non-isomorphicity of source and target syntax trees has proven problematic in practice, with naïve implementations severely underperforming compared to string-to-tree or tree-to-string models (Chiang, 2010). Chiang’s solution, extended by Zhang

et al. (2011), is to use syntax as a soft constraint. This improves translation quality, though the resulting model is somewhat more complex.

STSG was proposed as a suitable formalism for tree-to-tree translation by Eisner (2003). The grammar rules include tree fragments on both sides of the rules. For example, rules to derive the segmentation shown in Figure 2.7 can be written



(r4)



(r5)

2.5.5 Shortcomings of Syntax-based Models

Syntax-based models have primarily been motivated by the need to efficiently model the long-distance reorderings such as those found between SVO and SOV languages. The synchronous formalisms that have been employed allow the direct incorporation of syntactic information into the translation rules. However, the models make no direct provision for other forms of linguistic knowledge, such as morphology or semantics. Whilst the grammar labels are arbitrary and permit the inclusion of information beyond syntax, there has so far been little research in this area.

The dependence on linguistic resources and tools restricts most of the syntax-based approaches to the small number of languages in which those resources are available. The exception is Chiang's hierarchical phrase-based model, which requires no linguistic resources. Even where resources exist, the linguistic annotation may not be ideally suited to the task of translation. Addressing these mismatches is an active area of research, particularly in handling non-constituent phrases (Zollmann and Venugopal, 2006; Burkett and Klein, 2012).

2.6 Conclusion

In this chapter we presented the three main modelling approaches that are used in statistical machine translation: word-based, phrase-based, and syntax-based. Our main emphasis was on the linguistic expressiveness of the models. SCFG or STSG-based models can readily incorporate syntactic annotation in the form of non-terminal labels and recent models have incorporated phrase-structure and dependency labels .

At the time of writing, phrase-based and syntax-based models offer similar translation quality in empirical comparisons, usually with one or other appearing to have the edge for any particular language pair (Zollmann et al., 2008; Kalijahi et al., 2012). The exception is tree-to-tree, which typically underperforms due to syntactic-divergence, though approaches are being developed to address this issue (Chiang, 2010).

The small differences in automatically-measured translation quality together with differences in implementation details make it difficult to draw any firm conclusions that one approach is inherently superior to the other. However, syntax-based models offer greater scope for developing linguistically richer models and improving the integration of syntax is an active area of research.

Chapter 3

String-to-Tree Translation

3.1 Introduction

This chapter discusses string-to-tree models in greater depth, introducing the major algorithms used for rule extraction and decoding with these models. We first describe the string-to-string rule extraction algorithm of Chiang's (2005) Hiero hierarchical phrase-based model, which can straightforwardly be extended to incorporate syntactic annotation (Zollmann and Venugopal, 2006; Chiang, 2010). We then describe the alternative GHKM algorithm (Galley et al., 2004), which was designed specifically for string-to-tree models.

Decoding for these models involves a monolingual parsing step and we describe two parsing algorithms that have been used for translation, CYK+ (Chappelier and Rajman, 1998) and Hopkins and Langmead's (2010) chart parsing algorithm for scope-k grammars. Efficiently integrating m-gram language model scoring into syntax-based decoding has proven challenging; we describe cube pruning (Chiang, 2007), the most widely used approximate approach.

We will use a string-to-tree model as our baseline in experiments thoughout this thesis. In subsequent chapters we will adapt these rule extraction and decoding algorithms to incorporate unification-based constraints in the grammar and to enforce constraints during decoding.

3.2 Rule Extraction

Syntax-based rule extraction has largely developed along two lines, one originating in hierarchical phrase-based translation (Chiang, 2005; 2007) and the other in GHKM

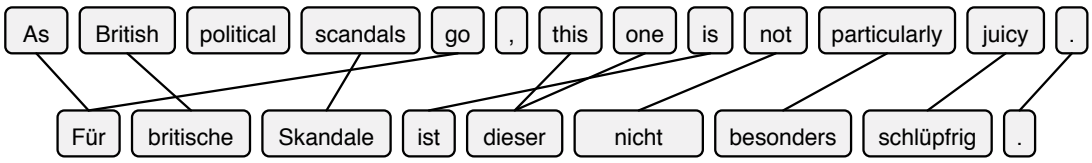


Figure 3.1: Word alignment from Figure 2.3

(Galley et al., 2004; 2006).

Hierarchical rule extraction generalizes the established phrase-based extraction method to produce formally-syntactic synchronous context-free grammar rules without any requirement for linguistic annotation of the training data. In subsequent work, the approach has been extended to incorporate linguistic annotation on the target side (as in SAMT (Zollmann and Venugopal, 2006)) or on both sides (Chiang, 2010).

In contrast, GHKM presupposes that parse trees are available for the target-side of the parallel corpus. It places target-side syntactic structure at the heart of the rule extraction process, producing STSG rules that map strings to tree fragments.

3.2.1 Hiero and Syntactic Extensions

The rule extraction algorithm in Chiang's hierarchical phrase-based model takes as input a parallel corpus of sentence pairs with many-to-many word alignments. It proceeds in two steps, which are repeated for every sentence pair. For examples, we use one of the word-aligned sentence pairs from Figure 2.3, which for convenience we repeat in Figure 3.1.

Step 1: Lexical Rules

An initial set of phrase-pairs is extracted from each word-aligned sentence pair $\langle s,t \rangle$ using the standard phrase-based approach (Koehn et al., 2003; Och and Ney, 2004). Informally, the initial set is the set of all substring pairs of the form $\langle \bar{s},\bar{t} \rangle$ in which at least one word of $\bar{s}$ is aligned to one or more words of $\bar{t}$ and in which no word of $\bar{s}$ is aligned to a word outside $\bar{t}$ (and vice-versa). Each phrase pair, $\langle \bar{s},\bar{t} \rangle$, forms a lexical grammar rule $\mathrm{X} \rightarrow \bar{s} \mid \bar{t}$.

For the sentence pair in Figure 3.1, the set of lexical rules includes:

$$\mathrm{X} \rightarrow \textit{scandals} \mid \textit{Skandale}$$

$$\begin{aligned} \text{X} &\rightarrow \textit{political scandals} \mid \textit{Skandale} \\ \text{X} &\rightarrow \textit{British political scandals} \mid \textit{britische Skandale} \end{aligned}$$

Step 2: Non-Lexical Rules

Having extracted the initial set of lexical rules for a sentence pair, non-lexical rules are generated by repeatedly choosing pairs of rules r_1 and r_2:

$$\text{X} \rightarrow \bar{s} \mid \bar{t} \qquad (r_1)$$

$$\text{X} \rightarrow \gamma \mid \alpha \qquad (r_2)$$

such that r_1 is a lexical rule from step 1, r_2 is a rule from either step, and $\bar{s}$ and $\bar{t}$ are substrings of γ and α, respectively. A new rule r_3 is formed from r_2 by substituting a pair of non-terminals for the substrings $\bar{s}$ and $\bar{t}$.

For our example sentence pair, this process produces the following rules, among others:

$$\begin{aligned} \text{X} &\rightarrow \text{X}_1 \textit{ Skandale} \mid \text{X}_1 \textit{ scandals} \\ \text{X} &\rightarrow \textit{für } \text{X}_1 \mid \textit{as } \text{X}_1 \textit{ go ,} \\ \text{X} &\rightarrow \textit{für } \text{X}_1 \textit{ ist } \text{X}_2 \textit{ nicht} \mid \textit{as } \text{X}_1 \textit{ go , } \text{X}_2 \textit{ is not} \end{aligned}$$

Limiting the Grammar Size

The algorithm as currently defined would extract an unusably large set of rules. To restrict grammar size, Chiang (2007) imposes the following limits:

1. Initial phrase pairs are discarded if they have unaligned words at the edges of phrases.
2. Initial phrase pairs are limited to 10 words on either side.
3. Rules are limited to five non-terminals plus terminals on the source side.
4. Rules are limited to two non-terminals.
5. Source-side non-terminals must not be adjacent.
6. Rules must include at least one pair of aligned words.

All of the example rules in this section fulfil these criteria.

Label	Condition
C	$\bar{t}$ is exhaustively dominated by a node with label C
X	$\bar{t}$ is not exhaustively dominated by any node
$C_1 + C_2$	$\bar{t}$ is exhaustively dominated by adjacent nodes with labels C_1 and C_2
C_1/C_2	$\bar{t} = t_i, \ldots, t_j$ and there exists a $k > j$ such that $t_i, \ldots, t_k$ is exhaustively dominated by a node with label C_1 and $t_{j+1}, \ldots, t_k$ is exhaustively dominated by a node with label C_2
$C_2 \backslash C_1$	$\bar{t} = t_i, \ldots, t_j$ and there exists a $h < i$ such that $t_h, \ldots, t_{i-1}$ is exhaustively dominated by a node with label C_1 and $t_i, \ldots, t_k$ is exhaustively dominated by a node with label C_2

Table 3.1: Labelling rules for SAMT

Adding Syntactic Annotation

Given parse trees for the source or target training sentences, Chiang's rule extraction method can be adapted to produce string-to-tree, tree-to-string, or tree-to-tree grammars. Zollmann and Venugopal (2006) take this approach to develop the string-to-tree Syntax-Augmented Machine Translation (SAMT) model.

In the SAMT model, non-terminal labels are derived from target-side parse tree labels. If a parse tree node with label C exhaustively dominates the target words of a lexical rule then the rule is given the label C instead of the generic X that would be used in Hiero. This raises the question of how to label the remaining lexical rules, or whether they should be extracted at all. The latter option risks discarding potentially useful rules and so Zollmann and Venugopal (2006) introduce a set of rules for generating complex labels. Step 1 of Chiang's method is thus adapted such that a lexical rule for an initial phrase pair $\langle \bar{s}, \bar{t} \rangle$ is assigned a label according to the rules given in Table 3.1. A phrase pair may satisfy multiple conditions, resulting in multiple rules with distinct labels.

Having labelled the initial rules, Zollmann and Venugopal (2006) apply Step 2 (without modification) to generate the full grammar.

For example, given the word-aligned sentence pair from Figure 3.1 and the parse tree in Figure 3.2, the SAMT algorithm would generate the following rules (among many others):

$$\text{ADJP/JJ} \rightarrow \textit{nicht besonders} \mid \textit{not particularly}$$

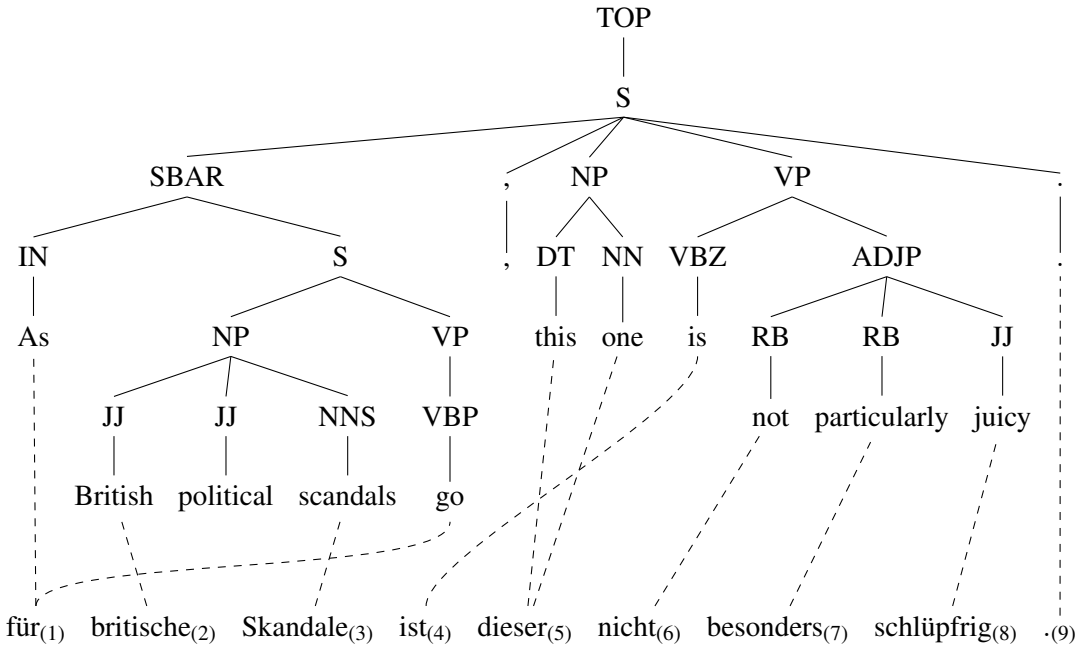


Figure 3.2: Word-aligned sentence pair with target-side parse tree

$$\begin{aligned}
\text{NP+VBZ} &\rightarrow \textit{ist dieser} \mid \textit{this one is} \\
\text{X} &\rightarrow \textit{ist dieser nicht} \mid \textit{this one is not} \\
\text{NP+VP} &\rightarrow \text{X}_1 \textit{ nicht besonders } \text{X}_2 \mid \text{NP+VBZ}_1 \textit{ not particularly } \text{JJ}_2
\end{aligned}$$

Whilst SAMT's extraction algorithm retains the high phrasal coverage of Hiero, the labelling scheme leads to a large set of non-terminals (typically numbering in the thousands), which can cause sparsity issues for parameter estimation and increase the grammar-related costs of decoding. Subsequent research has proposed rule labelling methods that lessen these issues whilst still producing labels for non-constituent target spans: Hanneman and Lavie (2013) introduce a technique to cluster SAMT labels, and Weese et al. (2012) use labels taken from CCG derivations.

3.2.2 GHKM

As we have seen, the Hiero rule extraction algorithm does not use or require linguistic annotation. Extensions that add syntactic annotation, such as SAMT, do so by generating labels for the rules; they do not take syntactic structure into account when determining which rules to extract. In contrast, Galley et al.'s (2004) GHKM algorithm assumes the availability of target-side parse trees for all sentence pairs. Given a word-aligned sentence pair and parse tree, GHKM generates a set of synchronous rules

that can be applied to the source sentence in order to reconstruct the target parse tree, something that is not necessarily possible with Hiero (assuming the usual restrictions on rule size). The algorithm is motivated in part by Fox's (2002) finding that early syntax models could not account for some of the complex reordering relationships that occur in human translation data.

In Figure 3.2, the word-aligned sentence pair was shown together with a target-side parse tree. GHKM treats the two as a single directed graph structure called an *alignment graph*. Given a word-aligned sentence pair and parse tree, the graph is formed in a straightforward manner: there are nodes for each of the source words, target words, and parse tree nodes, and there are edges for each of the word alignment links and parse tree edges. The graph edges are directed from the parse tree root node toward the target word nodes and from the the target word nodes toward the source word nodes.

The algorithm centres around the classification of parse tree nodes into two types, *frontier nodes* and non-frontier nodes. A frontier node can be thought of as being the head of a subtree with a yield $\bar{t}$ that is aligned exclusively to a single substring $\bar{s}$ of the source sentence (cf. the definition of SAMT's *C*-labelled lexical rules).

A few definitions are required before the algorithm can be presented. The *span* of a node n is the set of source word nodes reachable from n. For example, in Figure 3.2, the span of the VP node is $\{s_4, s_6, s_7, s_8\}$. The *closure* of a span is the smallest superset that contains a contiguous sequence of source word nodes. For the VP node, the closure is $\{s_4, s_5, s_6, s_7, s_8\}$. The *complement span* is defined recursively: for the root node it is the empty set; for a non-root node n it is the union of the complement span of n's parent with the spans of n's siblings. The complement span of the VP node is $\{s_1, s_2, s_3, s_5, s_9\}$. Note that the complement span is not necessarily the set-theoretic complement of the span. For example, the span and complement span of the lower S node both contain s_1.

The definition of a frontier node can now be given more precisely: a node n is a frontier node if the intersection of its closure and its complement span is empty. In other words, if the closure of n's span contains no source nodes that are reachable from nodes other than n or n's descendents. In Figure 3.2, the ADJP node is a frontier node, but its parent VP node is not (because its span's closure includes *dieser*, which is reachable from a non-descendent NP).

Figure 3.3 shows the algorithm. The input is an alignment graph g and the output is a list of STSG rules. It is assumed that the span of each node is already known. For a node n the corresponding span is referred to as $n.span$. The spans are used to compute

GHKM(g)

```
 1  rules = empty list
 2  for each node n in g, visited in topological order
 3      n.c-span = ∅
 4      for each parent p of n
 5          n.c-span = n.c-span ∪ p.c-span
 6      for each sibling s of n
 7          n.c-span = n.c-span ∪ s.span
 8  frontier-set = ∅
 9  for each node n in g, visited in topological order
10      if n.c-span ∩ CLOSURE(n.span) = ∅
11          frontier-set = frontier-set ∪ n
12  for each node n in frontier-set
13      define a subgraph h containing n, its children, and any connecting edges
14      while h contains a sink node s ∉ frontier-set
15          expand h such that it include s's children and any connecting edges
16      γ = sink nodes of h ordered by span position
17      π = h with sinks replaced in span position order by variables x_1 ... x_n
18      add rule (γ, π) to rules
19  return rules
```

Figure 3.3: The GHKM rule extraction algorithm

the complement spans (c-spans) of every node (lines 2-7). The spans and complement spans are used in line 10 to identify the frontier nodes. Lines 12–18 produce one rule from each frontier node.

The rules are *minimal* in the sense that the corresponding subgraphs are expanded only to the extent necessary to be consistent with the word alignments — they include as little context as possible. The combination of graphically-adjacent minimal rules into larger, contextually-richer rules (called *composed rules*) has been found to significantly improve translation quality (Galley et al., 2006).

3.2.3 Closing Remarks

Ultimately, both the Hiero and GHKM rule extraction algorithms define rules according to a sentence pair's word-alignments using the same notion of consistency that is carried through from phrase-based models. Without any restriction on rule size they will produce an exponentially large set of rules and so in practice only a subgrammar can be extracted. It is the differing rule selection heuristics that distinguish the two

approaches, with hierarchical approaches being motivated by phrasal coverage and GHKM by target-side tree coverage.

3.3 Decoding as Parsing

Recall from Section 2.5.1 that the objective of a syntax-based translation system is to find the synchronous derivation d^* with the highest total score according to a sum of weighted feature function scores (Equation 2.13). The target yield of d^* is the output translation, which is an approximation to the highest-scoring translation.

As Chiang (2007) notes, the search problem defined by Equation 2.13 could be solved exactly by a dynamic-programming algorithm if the calculation of a derivation d's weighted score could be decomposed into a sum of weighted scores for d's subderivations (in dynamic-programming terms, if the problem had *optimal substructure* and *overlapping subproblems*). Unfortunately, the m-gram language model score of d cannot be decomposed this way: the m-gram LM score of a subderivation over a substring $[i, j]$ cannot be fully calculated without knowledge of the target words produced by translating source words outside $[i, j]$ and therefore the optimality of a subderivation cannot be determined without knowledge of some larger optimal derivation.

We will return to the problem of integrating language model scoring later. For now, we note that the problem can be solved approximately through a combination of dynamic-programming and beam-search. As such, our search algorithm must record partial solutions (bounded in number by the beam width) at each step of the dynamic-programming procedure.

If we disregard language model scoring, translation is simply a variation on weighted monolingual parsing: we can parse the input sentence using a monolingual projection (described shortly) of our synchronous grammar to find the highest-scoring derivation and then recover the translations as a post-processing step. We can therefore employ the dynamic-programming algorithms that have been developed for monolingual parsing. The choice of parsing algorithm may be influenced by properties of the grammar, which will depend on the rule extraction method and whether binarization (of the tree or grammar) is used.

Variants of CYK (Kasami, 1965) are frequently used for decoding. We describe the CYK+ algorithm. Specialised chart parsing algorithms for translation grammars have been developed by DeNero et al. (2009) and Hopkins and Langmead (2010). We describe the latter, which is suited to a larger subclass of synchronous grammars.

Before presenting the parsing algorithms, we describe the projection that produces the monolingual grammar.

3.3.1 Projecting the Synchronous Grammar

As we have seen, string-to-tree translation rules use linguistically-motivated non-terminal labels on the target-side and the generic X non-terminal on the source-side. For the purposes of parsing-based decoding, the source-side of the synchronous grammar is used, with target-side non-terminal labels projected onto their corresponding source-side non-terminals. For example, the SCFG rule:

$$\text{SBAR} \rightarrow \textit{für}\ \text{X}_1 \ \mid\ \textit{As}\ \text{NP}_1\ \textit{go}$$

is projected to the CFG rule:

$$\text{SBAR} \rightarrow \textit{für}\ \text{NP}$$

This process produces a monolingual grammar from a synchronous grammar with a one-to-many mapping from monolingual to synchronous rules.

3.3.2 The CYK+ Parsing Algorithm

CYK+ (Chappelier and Rajman, 1998) is a close algorithmic relation of the better-known CYK and Earley (Earley, 1970) algorithms. It has the same $O(n^3)$ upper-bound, but removes CYK's restriction that the grammar must be in Chomsky Normal Form (for a discussion of why monolingual grammar binarization techniques are problematic for synchronous grammars, see Zhang et al. (2006)).

In Chappelier and Rajman's (1998) original description, the algorithm requires that grammar rules are either purely lexical or purely non-lexical, but they note that this restriction was made to simplify the algorithm and is easily eliminated. Since partially lexicalized rules are ubiquitous in translation grammars we describe a modified version here.

In addition to the standard $|s| \times |s|$ chart, CYK+ associates two lists with each chart cell. For each cell, chart$[i, j]$, there is a *type-1* list, which records the set of labels A for which $A \Rightarrow^* s_i \ldots s_j$, and a *type-2* list, which records the set of Earley-style *dotted rules* $\alpha\bullet$ such that $\alpha \Rightarrow^* s_i \ldots s_j$ and such that the grammar contains some rule $A \rightarrow \alpha\beta$, where α and β are non-empty strings of terminals and non-terminals.

Figure 3.4 shows the CYK+ algorithm. For readability, the pseudocode shows a CYK+ recogniser (the input s is recognised if the type-1 list of chart cell $[1, |s|]$ contains

```
CYK-PLUS-PARSE(G, s)
 1  for i = 1 to |s|                                          // initialize
 2      MATCH(G, i, i, s_i)
 3  for width = 1 to |s|
 4      for i = 1 to |s| − width + 1                          // start
 5          // "Standard" filling procedure
 6          j = i + width − 1                                 // end
 7          for k = i + 1 to j                                // split-point
 8              for α• in type-2 list of chart[i, k − 1]
 9                  for B in type-1 list of chart[k, j]
10                      MATCH(G, i, j, αB)
11          // Special handling for partially-lexicalized rules
12          if width > 1
13              for α• in type-2 list of chart[i, j − 1]
14                  MATCH(G, i, j, αs_i)
15          // "Self-filling" procedure
16          for B in type-1 list of chart[i, j]
17              MATCH(G, i, j, B)

MATCH(G, i, j, π)
1  for each rule A → πγ in G
2      if γ is empty
3          add A to type-1-list of chart[i, j]
4      else
5          add π• to type-2-list of chart[i, j]
```

Figure 3.4: The CYK+ algorithm.

the start symbol). As with CYK, the algorithm can straightforwardly be extended to build a representation of the parse forest or to find the Viterbi parse of a weighted grammar.

The algorithm visits chart cells in order of increasing width (for cells of the same width, the order is unimportant, but the algorithm shown visits them from left-to-right). At each cell, the algorithm attempts to extend predecessor dotted rules to cover the current span by adding a non-terminal symbol (lines 8-10) or a terminal (lines 12-13). Dotted rules that cover the current cell's full span with a single non-terminal are produced during the "self-filling" procedure in lines 15-16. This procedure also handles unary rule applications over the span.

3.3.3 The Scope-3 Parsing Algorithm

In general, context-free parsing complexity is exponential with respect to the arity (or rank) of the grammar (that is, the maximum number of non-terminals on any rule right-hand side), with cubic-time parsing relying on either explicit binarization (like in CYK (Kasami, 1965)) or implicit binarization (like in the Earley parser (Earley, 1970)). This is problematic for GHKM grammars, which have no inherent limit on rank: whilst effective synchronous binarization techniques have been developed (Zhang et al., 2006; Huang et al., 2009), they come with their own drawbacks, including inflation of the number of grammar rules and implementation complexity. With the specific, highly-lexicalised nature of translation grammars in mind, Hopkins and Langmead (2010) define a useful subclass, called scope-k grammars. They prove that a grammar with scope k can be used to parse a sentence of length n in $O(n^k)$ chart updates without binarization. They also show empirically that reducing a GHKM grammar to scope-3 by pruning does not harm translation quality compared to synchronous binarization.

In order to define the concept of scope and Hopkins and Langmead's (2010) chart parsing algorithm, we must first define a few concepts: an *application context* is an object describing the span over which a grammar rule is applied and the subspans to which its source non-terminals are applied. A *rule pattern* is the source side of a rule with non-terminal labels replaced by a special substitution symbol $\diamond$. A *label sequence* in general is a sequence of non-terminal labels. A rule's label sequence is its sequence of non-terminal labels in source-side order.

The concept of scope can now be defined: for a grammar rule r with pattern p, r's scope is the number of pairs of adjacent substitution symbols in the pattern $\diamond p \diamond$. For example,

Pattern	Scope	Pattern	Scope
a b c d e	0	a ◇ ◇ ◇ e	2
a ◇ c ◇ e	0	◇ b c d ◇	2
a ◇ ◇ d e	1	◇ ◇ c d ◇	3
◇ b c d e	1	◇ ◇ ◇ ◇ ◇	6

The maximum number of possible application contexts for a rule is a function of its scope and sentence length. The scope of a grammar $\mathcal{G}$ is the maximum scope of any rule in $\mathcal{G}$.

Figure 3.5 shows Hopkins and Langmead's (2010) parsing algorithm. It takes as input a scope-k grammar G and an input sentence s. Each chart cell $[i, j]$ contains a list

```
SCOPE-PARSE(G, s)
 1  PRE-COMPUTE-PATTERN-CONTEXT-PAIRS(G, s)
 2  for width = 1 to |s|
 3      for i = 1 to |s| − width + 1
 4          j = i + width − 1
 5          pairs = RETRIEVE-PATTERN-CONTEXT-PAIRS(i, j)
 6          for each pair (pattern, context) in pairs
 7              lhs-rhs-pairs = RETRIEVE-LABELS(pattern)
 8              for each (lhs, rhs in lhs-rhs-pairs
 9                  check for rhs's labels in cells defined by context, continue if missing
10                  add lhs to chart[i, j]
```

Figure 3.5: The scope parsing algorithm

of non-terminal labels from the left-hand-side of rules that cover the span (identical to CYK+'s type-1 list).

At line 1, the complete set of $(pattern, application\ context)$ pairs is generated. That is, if a grammar rule has a pattern p that can be applied to the input sentence with application context c, the pair (p, c) is recorded. The pairs are stored for lookup according to chart cell position (line 5). Line 7 retrieves the set of pairs (lhs, rhs) for all rules in the grammar that share a specific pattern, where *lhs* is the left-hand-side label of the rule and *rhs* is the label sequence of the rule. These sets can be pre-computed once per grammar.

Williams and Koehn (2012) provides details of how to efficiently implement the steps at lines 1 and 9.

For readability, the pseudocode shows a recogniser (the input s is recognised if the chart cell $[1, |s|]$ contains the start symbol. As with CYK+, the algorithm can be extended to build a representation of the parse forest or to find the Viterbi parse of a weighted grammar.

3.3.4 The Parse Hypergraph

Parsing an input sentence with a context-free grammar results in a forest of parse trees. As Klein and Manning (2001) show, the parse forest can be represented by a directed hypergraph, a generalisation of a directed graph in which a hyperedge can connect two sets of nodes. Hypergraphs prove to be useful for reasoning about and implementing syntax-based decoding algorithms since the search space can also be represented as

a hypergraph. A representation of the parse forest, which we refer to as the parse hypergraph, is used as a starting point for exploring the decoding search space.

More specifically, a context-free parse forest can be represented by a hypergraph that is rooted, connected, labelled, directed, and acyclic. The source nodes of the hypergraph correspond to input words and the non-source nodes to parsing states, with the root node representing the goal state. Each hyperedge corresponds to a derivation step, connecting one or more parsing states to a successor parsing state via the application of a rule.

Each rule in a projected monolingual grammar maps to one or more rules in the synchronous grammar, therefore each hyperedge in the parse hypergraph can be thought of as representing one or more synchronous rule applications. The hypergraph, in combination with the grammar projection, thus represents the full space of synchronous derivations for an input sentence.

Figure 3.6 gives an example SCFG grammar, which has the monolingual CFG projection shown in Figure 3.7. Figure 3.8 shows the hypergraph representation of the parse forest formed by parsing the input sentence 'für britische Skandale' with the projected monolingual grammar.

Depending on the input length, n, and on properties of the projected grammar G, it may be practical to construct the entire parse hypergraph: for an arbitrary scope-3 grammar, the parse hypergraph contains a maximum of $O(n^3|G|)$ hyperedges, since scope-3 pruning guarantees[1] that the maximum number of application contexts for any rule is $O(n^3)$. The maximum number of nodes is $O(n^2|C|)$, where C is the set of non-terminals.

3.3.5 The Search Hypergraph

As we have already noted, an exact search for the highest scoring synchronous derivation cannot be performed by a dynamic programming approach that decomposes the problem among subderivations because the m-gram language model must score sequences of target words that cross subderivation boundaries. For the same reason, the locally-optimal subderivation at a given node in the (weighted) parse hypergraph is not guaranteed to include the highest-scoring incoming hyperedge, nor is it guaranteed to include subhypergraphs that are themselves locally-optimal.

[1] Under the assumption that each word occurs only once in a given input sentence. See Hopkins and Langmead (2010) for a discussion of this assumption.

$$\begin{aligned}
\text{SBAR} &\rightarrow \textit{für}\ \text{X}_1 \mid \textit{As}\ \text{NP}_1\ \textit{go} && \text{(r1)}\\
\text{NP} &\rightarrow \textit{britische Skandale} \mid \textit{British political scandals} && \text{(r2)}\\
\text{NP} &\rightarrow \textit{britische Skandale} \mid \textit{British scandals} && \text{(r3)}\\
\text{NP} &\rightarrow \text{X}_1\ \text{X}_2 \mid \text{JJ}_1\ \text{NNS}_2 && \text{(r4)}\\
\text{JJ} &\rightarrow \textit{britische} \mid \textit{British} && \text{(r5)}\\
\text{JJ} &\rightarrow \textit{britische} \mid \textit{UK} && \text{(r6)}\\
\text{NNS} &\rightarrow \textit{Skandale} \mid \textit{scandals} && \text{(r7)}
\end{aligned}$$

Figure 3.6: SCFG grammar fragment.

$$\begin{aligned}
\text{SBAR} &\rightarrow \textit{für}\ \text{NP} && \text{(q1)}\\
\text{NP} &\rightarrow \textit{britische Skandale} && \text{(q2)}\\
\text{NP} &\rightarrow \text{JJ}\ \text{NNS} && \text{(q3)}\\
\text{JJ} &\rightarrow \textit{britische} && \text{(q4)}\\
\text{NNS} &\rightarrow \textit{Skandale} && \text{(q5)}
\end{aligned}$$

Figure 3.7: Monolingual projection of SCFG grammar in Figure 3.6

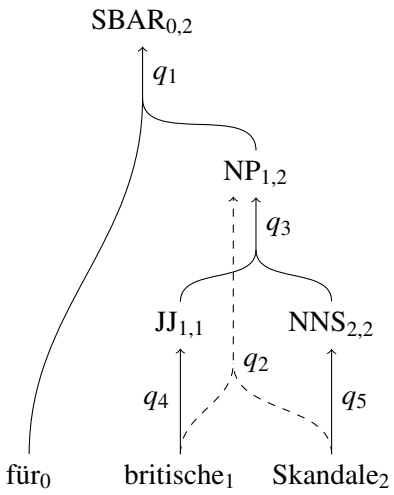


Figure 3.8: Hypergraph representing the parse forest generated by parsing the input 'für britische Skandale' with the grammar in Figure 3.7. There are two derivations: the first is indicated using solid arrows. The second differs in the production of the NP, as indicated by dashed lines.

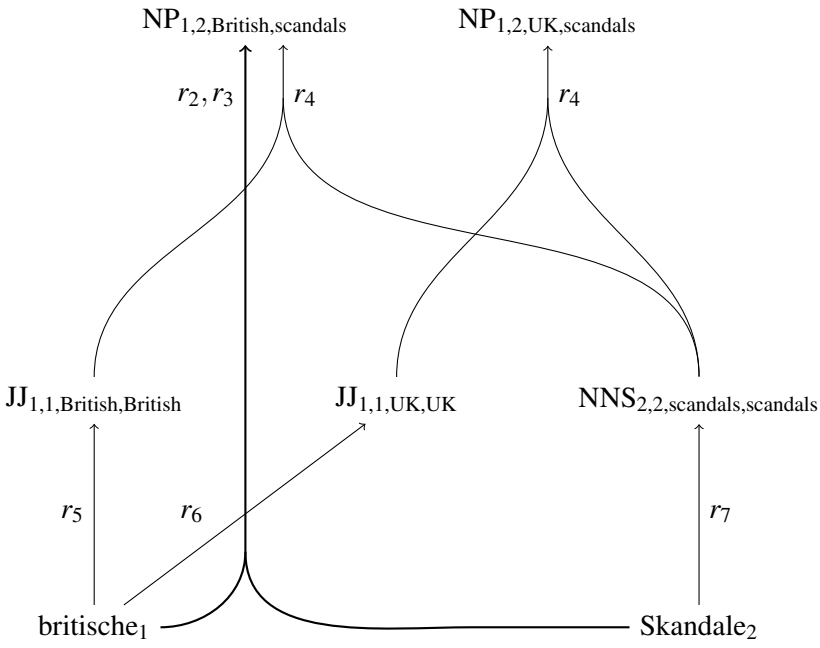


Figure 3.9: Search hypergraph representing the portion of the search space corresponding to the second two words of the input 'für britische Skandale' and the grammar in Figure 3.6.

However, the parse hypergraph can in principle be expanded to a larger hypergraph, the search hypergraph, in which these properties do hold: whereas a non-leaf node in the parse hypergraph represents the set of subderivations that share a category and span, the non-leaf nodes in the search hypergraph must also differentiate (some of) the target words generated during translation since those are relevant to m-gram scoring of superderivations. Specifically, each node must record the $m-1$ words to the left of the translation and the $m-1$ words to the right. Since the target words are relevant to search, the hyperedges must now correspond to rules in the original grammar, not the projected grammar.

Figure 3.9 shows an example search hypergraph that expands (part of) the parse hypergraph from Figure 3.8. It assumes a bi-gram language model and therefore records one left boundary word and one right boundary word.

Using dynamic programming, the optimal translation can be determined from the search hypergraph in time linear to the size of the hypergraph. Unfortunately, exploring the full search hypergraph is usually impractical. The maximum number of hyperedges is $O(n^3|G||T|^{2A(m-1)})$ where T is the set of target-side terminals and A is the maximum rule arity (the number of non-terminals in the right-hand-side). The number of nodes

```
CUBE-PRUNE(n, k)
 1  beam = empty ordered list
 2  q = empty priority queue
 3  for e in n's hyperedge set
 4      PUSH(q, CUBE-PRUNE-HYPEREDGE(e))
 5  while |beam| < k and |q| > 0
 6      lazy-list = POP(q)
 7      (hypothesis, score) = POP(lazy-list)
 8      ADD-TO-BEAM(beam, (hypothesis, score))
 9      if lazy-list is not empty
10          PUSH(q, lazy-list)
11  return beam

CUBE-PRUNE-HYPEREDGE(e)
1  q = empty priority queue
2  h = BEST-FIRST-HYPO(e)
3  PUSH(q, (h, SCORE(h)))
4  while |q| > 0
5      hypothesis = POP(q)
6      yield hypothesis
7      for h in CREATE-NEIGHBOURS(hypothesis)
8          PUSH(q, (h, SCORE(h)))
```

Figure 3.10: The cube pruning algorithm

is $O(n^2|C||T|^{2(m-1)})$.

3.3.6 Integrating m-gram Language Model Scores

To make the problem tractable, several approximate search algorithms have been proposed, the most widely used being cube pruning (Chiang, 2007), which we will describe shortly. All of the approximate algorithms involve a beam search where a limited set of subderivations is fully scored at each node of the parse hypergraph. The nodes of the parse hypergraph are visited in bottom-up order and at each node the beam is filled by combining and scoring subderivations from the beams of incoming nodes using the synchronous rules corresponding to incoming edges.

Cube Pruning

The cube pruning algorithm is shown in Figure 3.10. The function CUBE-PRUNE

operates on a hypergraph node n, filling a beam with hypotheses to a maximum size of k. A hypothesis is a subderivation together with auxiliary information. The function CUBE-PRUNE-HYPEREDGE is a generator function that lazily produces a sequence of scored hypotheses for a hyperedge e.

CUBE-PRUNE-HYPEREDGE initially constructs the single most promising hypothesis (line 2). This is the subderivation formed by combining the highest-scoring subderivations from the incoming nodes' beams using the highest-scoring synchronous rule. This initial hypothesis is scored using the model (including the language model score) and pushed onto a priority queue that holds hypothesis-score pairs in best-first order. The loop at lines 4-8 generates hypotheses by popping the highest scoring hypothesis then constructing its neighbouring hypotheses: the set of hypotheses that are identical except for either using the next best rule or substituing the next best hypothesis from one of the incoming nodes' beams.

CUBE-PRUNE begins by forming a lazy-list for each hyperedge in e's hyperedge set and pushing them onto a priority queue (line 4). This queue orders lists by the score of the hypothesis at the head of the list. Lists are then popped from the queue (line 6), the head hypothesis is removed from the list (line 7) and added to the beam (line 8) and then if the list still has items, it is pushed back onto the queue (line 10).

Alternatives to Cube Pruning

Cube pruning is currently the most widely-used algorithm for approximately exploring the search hypergraph. Alternatives include cube growing (Huang and Chiang, 2007), which seeks to reduce the search effort by employing a top-down approach that fills beams on demand instead of generating a fixed number of hypotheses at each node. The authors apply cube growing to tree-to-string decoding and Xu et al., (2012) apply it to string-to-string. We are not yet aware of any implementation for the string-to-tree case.

Heafield et al. (2013) presents an algorithm that exploits the tendency for the language model to score hypotheses more similarly if they share boundary words. At each node, partially-scored hypotheses with common boundary words are grouped according to the outer boundary words and the groups are lazily explored by progressively uncovering words and updating scores. Hypotheses are added to the beam as they are fully scored. The speed versus accuracy trade-off is shown empirically to be better than that of cube pruning for string-to-string and string-to-tree systems.

3.4 Conclusion

This chapter has introduced the main algorithms used for rule extraction and decoding in string-to-tree models. Our baseline model, which we will describe in detail in Chapter 6, uses the GHKM rule extraction algorithm with scope-3 pruning. Decoding uses Hopkins and Langmead's chart parsing algorithm with cube pruning for language model integration. We will later adapt the rule extraction and decoding algorithms from this chapter to incorporate unification-based constraints in the grammar and enforce constraints during decoding.

Chapter 4

Unification-based Approaches to Grammar

4.1 Introduction

In Chapter 2, we saw the basic unit of translation progress from words to phrases and then, in the syntax-based models, to hierarchical phrases. The last of these approaches was motivated primarily by the need to model reordering, since the word order of the source and target language can vary greatly, even when translating closely-related languages.

In SCFG and STSG grammars – the formalisms with which hierarchical phrases were defined – both the terminals and non-terminals are atomic symbols. In models that use linguistically motivated non-terminal labels, the labels are typically derived from treebank constituent labels. However, at the start of Chapter 2 we also saw examples of morphosyntactic relations, such as subject-verb agreement and case control, that fall outside the domain of constituency. In those two examples, the inflection expressed on one or more words must be consistent with respect to underlying linguistic properties. In practice, violations of this type of morphosyntactic relationship are common in machine translation output. This is especially true for morphologically-rich target languages where there can be inflectional distinctions that are not present in the source language.

In computational and theoretical linguistics, many monolingual models of phrase-structure grammar go beyond constituency by making use of *feature structures* to represent underlying linguistic properties of words and constituents. For example, in Functional Grammar (Kay, 1979) and Head-Driven Phrase Structure Grammar (Pol-

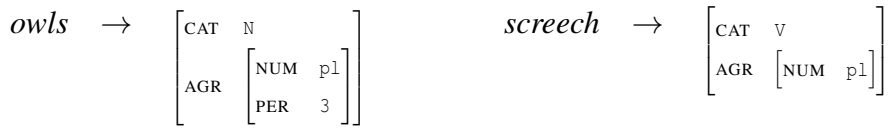


Figure 4.1: AVM representations for *owls* and *screech*

lard and Sag, 1994), feature structures encode phonological, syntactic, and semantic properties of words and phrases and even encode the grammatical rules. In Lexical-Functional Grammar (Bresnan, 1982), constituent structure (called c-structure) is defined in terms of words and constituent labels, which are atomic symbols as in context-free grammar, while a second, parallel, layer of structure, f-structure, uses feature structures to represent attributes such as grammatical function (subject, object, modifier, etc.) and agreement features.

Since their introduction at the end of the 1970s, feature structures and unification have proven powerful tools for modelling many aspects of natural language, enabling concise accounts of agreement, case control, verb subcategorization, and verb-raising, among others.

There is extensive literature on unification-based approaches to grammar, employing a rich variety of terminology and linguistic machinery. In this thesis, we use only a few of the core ideas, which we outline in this chapter. For simplicity of exposition and implementation, we borrow the terminology and notation of PATR-II (Shieber, 1984; 1986), a minimal unification-based formalism that extends context-free grammar. Our presentation in this chapter is kept informal. For a rigorous theoretical treatment of the topic, see Francez and Wintner (2011).

4.2 Feature Structures and Unification

Feature structures come in two varieties: *atomic* feature structures are untyped, indivisible values, such as `NP`, `nom`, or `sg`, and *complex* feature structures are partial functions that map features to values, with the values themselves being feature structures.

Conventionally, complex feature structures are written as attribute-value matrices (AVMs). Figure 4.1 shows two possible AVMs that represent category and agreement attributes for the noun *owls* and the verb *screech*.

A value within a complex feature structure can be specified using a path notation that describes the chain of features in enclosing feature structures. For both of the

examples in Figure 4.1, the path $\langle$AGR NUM$\rangle$ specifies the atomic value pl. If the symbol X refers to the feature structure for *owls* then we can write X($\langle$AGR PER$\rangle$) to refer to the value 3. This is also written $\langle$X AGR PER$\rangle$ when the meaning is clear from the context.

Values may be shared between features, either within the same complex feature structure or between multiple distinct feature structures. This is called *re-entrancy* and is denoted in AVMs using co-index boxes.

In order to describe the unification operation we first introduce the binary subsumption relation. One feature structure is said to *subsume* another if it contains (only) a subset of the information contained in the other. For example, the feature structure

$$\begin{bmatrix} \text{CAT} & \text{V} \end{bmatrix} \tag{X_1}$$

subsumes both itself and the more specific

$$\begin{bmatrix} \text{CAT} & \text{V} \\ \text{AGR} & \begin{bmatrix} \text{NUM} & \text{pl} \end{bmatrix} \end{bmatrix} \tag{X_2}$$

But it does not subsume either

$$\begin{bmatrix} \text{CAT} & \text{N} \\ \text{AGR} & \begin{bmatrix} \text{NUM} & \text{pl} \end{bmatrix} \end{bmatrix} \tag{X_3}$$

or

$$\begin{bmatrix} \text{AGR} & \begin{bmatrix} \text{NUM} & \text{pl} \end{bmatrix} \end{bmatrix} \tag{X_4}$$

Note that X_3's informational content is inconsistent with that of X_1 (since they contain conflicting CAT values), whereas X_4's is not. The respective content of X_1 and X_4 is able to coexist within a single feature structure and the smallest such feature structure is said to be their *unification*. The unification of X_1 and X_4 is in fact X_2. In general, the unification of two feature structures X and Y is the smallest feature structure Z that is subsumed by both X and Y, if such a feature structure exists.

The symbol $\sqsubseteq$ is used to denote the subsumption relation and the symbol $\sqcup$ is used to denote the unification operation. So we can write $X_1 \sqsubseteq X_2$ and $X_1 \sqcup X_4 = X_2$.

4.2.1 Graph-Based Unification Algorithms

An equivalent means of representating a feature structure, and the one typically used for implementation, is that of a rooted, labelled, directed graph. In this section, we describe a graph representation and the associated unification algorithm from Wroblewski (1987), which is adapted from Pereira (1985).

Each node of the graph is labelled with a record, `r`, containing three fields: `fwd`, `label`, and `feats`. The `fwd` field contains an auxiliary pointer that optionally forwards incoming edges onto a second node (otherwise it takes the special value `null`); the `label` field takes the special value `null` for complex feature structures, otherwise it holds a label representing the atomic value; finally, the `feats` field contains a set of edge pointers, which are labelled with the feature names.

Figure 4.2 shows graphs corresponding to the AVMs from Figure 4.1.

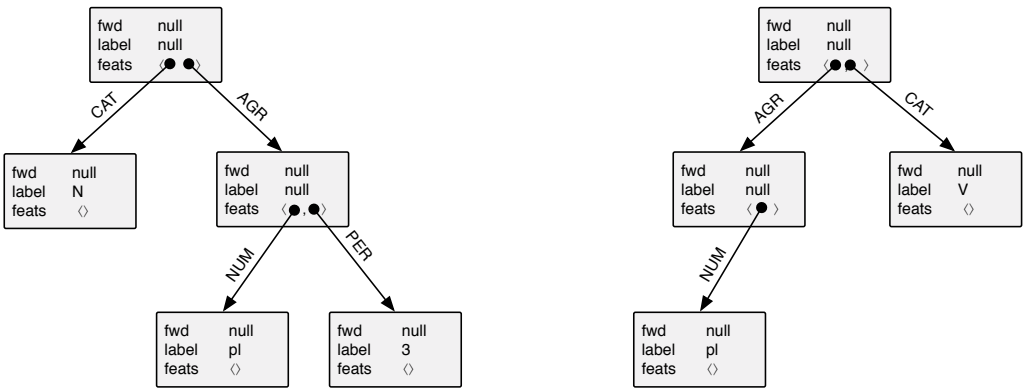


Figure 4.2: Graph representations of the feature structures in Figure 4.1

The graph unification algorithm is shown in Figure 4.3. The algorithm takes two feature structure graphs, f and g, as input and returns TRUE or FALSE, respectively, depending on whether the unification operation succeeded or not. On success, the contents of the input structures are altered such that they both describe the common unified value. On failure, the contents of the input structures are undefined. The algorithm is recursive (line 17). There are a number of ancillary functions: DEREFERENCE(n) returns n if $n.fwd$ is NULL or DEREFERENCE($n.fwd$) otherwise. All other functions use DEREFERENCE and operate on the end-point nodes. The IS-EMPTY, IS-COMPLEX, and IS-ATOMIC are simple predicate functions that are used as a shorthand for expressions that test the properties of their argument node. The VAL function returns the edge pointer that is labelled with the given feature name. The FIND-OR-CREATE-EMPTY function does likewise, except that it first tests for the existence of an outgoing edge from g with the given feature label. If no such edge exists, it creates an empty feature structure and adds an outgoing edge to g.

Figure 4.4 shows the modified graph structures from Figure 4.2 after applying the unification algorithm to the AGR feature structures.

```
UNIFY(f, g)
 1  f' = DEREFERENCE(f)
 2  g' = DEREFERENCE(g)
 3  if f' = g'
 4        return TRUE
 5  if IS-EMPTY(f)
 6        f.fwd = g'
 7        return TRUE
 8  if IS-EMPTY(g)
 9        g.fwd = f'
10        return TRUE
11  if (IS-ATOMIC(f) and IS-COMPLEX(g)) or (IS-ATOMIC(g) and IS-COMPLEX(f))
12        return FALSE
13  if IS-COMPLEX(f)
14        for feature in f'.feats
15              x = VAL(f, feature)
16              y = FIND-OR-CREATE-EMPTY(g, feature)
17              if UNIFY(x, y) = FALSE
18                    return FALSE
19  elseif f'.label ≠ g'.label
20        return FALSE
21  f.fwd = g'
22  return TRUE
```

Figure 4.3: Destructive graph unification algorithm

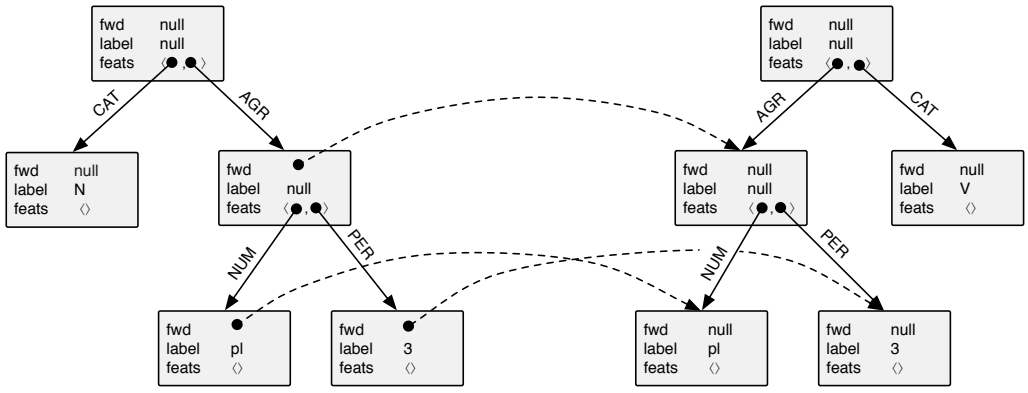


Figure 4.4: Feature structures from Figure 4.2 after unification of their AGR values.

Written as AVMs, the two modified input structures are:

$$owls \rightarrow \begin{bmatrix} \text{CAT} & \text{N} \\ \text{AGR} & \boxed{1} \end{bmatrix} \qquad screech \rightarrow \begin{bmatrix} \text{CAT} & \text{V} \\ \text{AGR} & \boxed{1}\begin{bmatrix} \text{NUM} & \text{pl} \\ \text{PER} & 3 \end{bmatrix} \end{bmatrix}$$

4.3 Grammar Rules

Grammar rules in PATR-II are a generalization of context-free grammar rules, with a single symbol on the left-hand side that rewrites to a sequence of symbols on the right. Whereas the symbols in context-free grammar are atomic, the non-terminal symbols in PATR-II are feature structure values. The rule symbols are denoted X_i where the subscript i indicates the symbol's position in the rule: X_0 is used for the left-hand side symbol and $\text{X}_1, \ldots, \text{X}_n$ for the right-hand-side symbols.

Each rule is associated with a, possibly empty, set of identities. These act as constraints on the values of the rule's feature structures. An identity relates two feature structure terms f and g, where f is $\text{X}_i(\pi)$ for some rule symbol X_i and feature path π. g is either a constant value or is $\text{X}_j(\rho)$ for some rule symbol X_j and feature path ρ.

A context-free grammar rule can be converted to a PATR-II rule by encoding the terminal and non-terminal labels as atomic feature structure values and identifying the values with the corresponding rule symbol values. Any context-free grammar is therefore trivially equivalent to a PATR-II grammar. For example, the context-free rules S $\rightarrow$ NP VP and N $\rightarrow$ *owl* could be written

$$\begin{array}{ll} \text{X}_0 \rightarrow \text{X}_1\ \text{X}_2 & \text{X}_0 \rightarrow \text{X}_1 \\ \quad \langle \text{X}_0 \rangle = \texttt{S} & \quad \langle \text{X}_0 \rangle = \texttt{N} \\ \quad \langle \text{X}_1 \rangle = \texttt{NP} & \quad \langle \text{X}_1 \rangle = \texttt{owl} \\ \quad \langle \text{X}_2 \rangle = \texttt{VP} & \end{array}$$

A PATR-II grammar becomes more interesting when complex feature structures are used as rule elements. The context-free constituency constraints can be retained by encoding category labels using an atomic feature within a richer feature structure. This feature can be given any name, but conventionally the name CAT is used. The CAT feature may be one of multiple features. For example, we could add a LEX feature for lexical values and an AGR feature to define agreement constraints on number and person values:

$$
\begin{array}{ll}
\mathrm{X}_0 \rightarrow \mathrm{X}_1\ \mathrm{X}_2 & \mathrm{X}_0 \rightarrow \mathrm{X}_1 \\
\quad \langle \mathrm{X}_0\ \textsc{cat} \rangle = \mathtt{S} & \quad \langle \mathrm{X}_0\ \textsc{cat} \rangle = \mathtt{N} \\
\quad \langle \mathrm{X}_1\ \textsc{cat} \rangle = \mathtt{NP} & \quad \langle \mathrm{X}_1\ \textsc{lex} \rangle = \mathtt{owls} \\
\quad \langle \mathrm{X}_2\ \textsc{cat} \rangle = \mathtt{VP} & \quad \langle \mathrm{X}_0\ \textsc{agr number} \rangle = \mathtt{pl} \\
\quad \langle \mathrm{X}_1\ \textsc{agr} \rangle = \langle \mathrm{X}_2\ \textsc{agr} \rangle & \quad \langle \mathrm{X}_0\ \textsc{agr person} \rangle = \mathtt{3}
\end{array}
$$

Note that the rule on the right uses identities to express the same information as the feature structure in Figure 4.1.

As a notational convenience, the CAT value of a rule element may substitute for the X symbol, allowing the corresponding CAT identity to be omitted:

$$
\begin{array}{ll}
\mathrm{S} \rightarrow \mathrm{NP}\ \mathrm{VP} & \mathrm{N} \rightarrow \textit{owls} \\
\quad \langle \mathrm{NP}\ \textsc{agr} \rangle = \langle \mathrm{VP}\ \textsc{agr} \rangle & \quad \langle \mathrm{N}\ \textsc{agr number} \rangle = \mathtt{pl} \\
 & \quad \langle \mathrm{N}\ \textsc{agr person} \rangle = \mathtt{3}
\end{array}
$$

The optionality of the CAT feature (or an equivalent) provides additional expressivity for the grammar author. Shieber (1986) gives an example of modelling verb subcategorization where forty similar rules can be collapsed into a single rule by allowing one of the constituents to take an underspecified category label. The requirement or otherwise of context-free constituency constraints varies between formalisms. For example, LFG requires that constituency structure is fully specified by the rules.

4.4 An Example Grammar Fragment

Figures 4.5 and 4.6 show the lexicon[1] and rules of a simple grammar *G*. The language defined by this grammar includes, for example, the sentence 'the owl screeches,' but not 'the owl screech,' in accordance with the English subject-verb agreement rules found in most dialects.[2]

Figure 4.7 shows a parse tree for a derivation of the sentence 'the owl screeches.' The non-leaf nodes show feature structures formed by satisfying the constraints through destructive unification, the result being that a single agreement value is shared between the noun, the verb, and the noun phrase. In a bottom-up interpretation, the NUMBER and PERSON values can be thought of as originating in the lexicon and being propagated upwards by constraint evaluation.

[1] Note that the person value $\neg 3$ is an atomic symbol. Whilst some unification-based formalisms allow logical operations in feature structures, we do not use them in this work.

[2] *G* does not allow for the interpretation of the latter as a noun phrase, though of course English does.

$$
\begin{array}{llll}
\textit{the} \rightarrow \begin{bmatrix} \text{CAT} & \text{DET} \\ \text{AGR} & \begin{bmatrix} \text{PER} & 3 \end{bmatrix} \end{bmatrix} & & \textit{an} \rightarrow \begin{bmatrix} \text{CAT} & \text{DET} \\ \text{AGR} & \begin{bmatrix} \text{NUM} & \text{sg} \\ \text{PER} & 3 \end{bmatrix} \end{bmatrix} \\
\textit{owl} \rightarrow \begin{bmatrix} \text{CAT} & \text{N} \\ \text{AGR} & \begin{bmatrix} \text{NUM} & \text{sg} \\ \text{PER} & 3 \end{bmatrix} \end{bmatrix} & & \textit{owls} \rightarrow \begin{bmatrix} \text{CAT} & \text{N} \\ \text{AGR} & \begin{bmatrix} \text{NUM} & \text{pl} \\ \text{PER} & 3 \end{bmatrix} \end{bmatrix} \\
\textit{screeches} \rightarrow \begin{bmatrix} \text{CAT} & \text{V} \\ \text{AGR} & \begin{bmatrix} \text{NUM} & \text{sg} \\ \text{PER} & 3 \end{bmatrix} \end{bmatrix} & & \textit{screech} \rightarrow \begin{bmatrix} \text{CAT} & \text{V} \\ \text{AGR} & \begin{bmatrix} \text{NUM} & \text{sg} \\ \text{PER} & \neg 3 \end{bmatrix} \end{bmatrix} \\
\textit{screech} \rightarrow \begin{bmatrix} \text{CAT} & \text{V} \\ \text{AGR} & \begin{bmatrix} \text{NUM} & \text{pl} \end{bmatrix} \end{bmatrix} & &
\end{array}
$$

Figure 4.5: Example grammar G (lexicon)

$$
\begin{array}{ll}
\text{S} \rightarrow \text{NP V} & \text{NP} \rightarrow \text{DET N} \\
\quad \langle \text{NP AGR} \rangle = \langle \text{V AGR} \rangle & \quad \langle \text{NP AGR} \rangle = \langle \text{DET AGR} \rangle \\
 & \quad \langle \text{NP AGR} \rangle = \langle \text{N AGR} \rangle
\end{array}
$$

Figure 4.6: Example grammar G (rules)

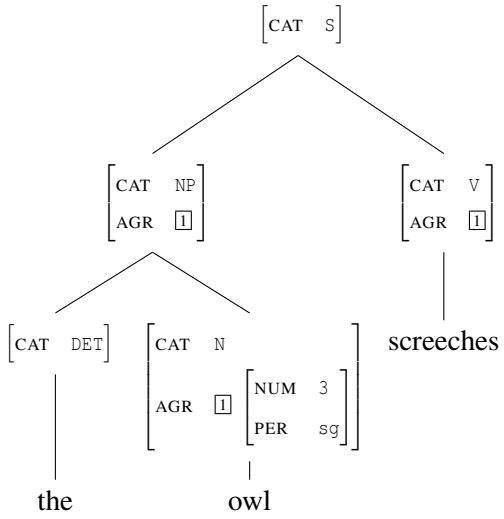


Figure 4.7: Parse tree for the sentence ‘the owl screeches’ as derived from grammar G

4.5 Unification-based Approaches to MT

Machine translation was proposed early on as an application for unification-based approaches, with Kay (1984) arguing that Functional Unification Grammar (FUG) would provide the expressive power to describe the conventional analysis, transfer, and generation steps of machine translation within a single formalism. Kay proposed that analysis and generation grammars for the source and target languages be bridged with a transfer grammar, and that all three could be expressed within the same unification-based framework.

4.5.1 Deep-Syntactic Transfer-Based Systems

The modern transfer-based models of Riezler and Maxwell (2006), Bojar and Hajič (2008), and Graham et al. (2009) are partial realizations of Kay's proposal, though they all take a looser approach to transferring the feature structures that are used for intermediate representation.

In these models, the source and target training sentences are first parsed using a unification-based parser (Riezler and Maxwell (2006) and Graham et al. (2009) both use LFG and Bojar and Hajič (2008) use Functional Generative Description). The resulting feature structures encode deep syntactic structure in the form of dependency relations between the elements of the sentence. Transfer rules between source and target feature structures are then learned and incorporated into a log-linear model with feature functions similar to those used in conventional phrase-based models. Having transferred the deep structural representations, the target side grammar is used to generate constituent structure and strings.

These models have so far been severely limited by the poor coverage of the non-stochastic parsers that are used, though Riezler and Maxwell (2006) find that their model compares favourably to a phrase-based model when the comparison is limited to in-coverage test data.

4.5.2 Stat-XFER

The Stat-XFER transfer-based framework (Lavie, 2008) is a tree-to-tree synchronous formalism with unification-based constraints that transfer information from source feature structures to target feature structures. The formalism allows source-side constraints for determining where transfer rules apply; target-side constraints for encour-

aging well-formedness during generation; and source-target constraints for transferring feature values.

Compared to the deep-syntactic transfer-based systems, Stat-XFER assumes less about the underlying linguistic theory. The framework is neutral with regard to the rule acquisition method and Lavie (2008) describes a manually developed Hebrew-English transfer grammar, which includes a small number of agreement feature transfer rules. In Hanneman et al. (2009) the framework is used with a large automatically-extracted grammar, though this does not incorporate non-constituent features.

4.6 Conclusion

This chapter has introduced the basic concepts of unification-based approaches to grammar. In this thesis we propose that targeted use of unification-based methods can be used to improve the grammaticality of statistical machine translation. In the next chapter we will introduce the framework that we use for exploring this approach. Our framework supports the use of target-side constraints in a string-to-tree statistical machine translation model.

Unification-based approaches have been applied to machine translation before, notably in transfer-based approaches where feature structures are deployed as the basic units for mapping information between the source and target languages. The transfer-based models typically employ linguistically-rich models for monolingual analysis and synthesis, but incorporate concepts from statistical machine translation into the transfer process. In a sense, we are working in the other direction, starting from minimally-linguistic statistical models and gradually introducing linguistically-motivated features and constraints.

Chapter 5

Framework

5.1 Introduction

This chapter presents the unification-based framework that we use throughout the rest of the thesis. Our framework is an extension of a conventional string-to-tree model that adds unification-based constraints to the target-side of the synchronous grammar rules. We describe the extended grammar formalism and how constraint evaluation is integrated into the search process.

The grammar extraction method is not prescribed by our framework since the definition of the constraints is dependent on the linguistic phenomena being modelled. In principle, a grammar extraction method could incorporate arbitrarily complex phenomena-specific processing to generate the constraints. In Chapter 7 we present one possible method for extracting rules with simple agreement constraints. This involves the automatic annotation of target trees together with a simple extension of GHKM. The method is then adapted to generate the constraints for a different application — verbal complex translation — in Chapter 8.

In the examples given in this chapter we use English as the target language since the linguistic phenomena will already be familiar to any reader. In subsequent chapters we will use German since there are prominent machine translation issues for which a unification-based approach seems well suited, but in Chapter 10 we will outline some potential applications in other languages, including English.

5.2 Formalism

Our model is an extension of the SCFG-based string-to-tree model described in Section 2.5.2. The formalism extends SCFG by adding a lexicon, which associates target-side terminals with feature structure values, and by adding target-side constraints to the rules.

5.2.1 Lexicon

Let T denote the target-side terminal vocabulary of a grammar. We associate with every terminal $t \in T$ a non-empty set of complex feature structures. This mapping is referred to as the lexicon. Our feature structures take the form described in Section 4.2 and we will list lexicon entries in the same style as we did there. For example, the entry for the terminal *screech* might be a set of two feature structures:

$$\textit{screech} \rightarrow \begin{bmatrix} \text{CAT} & \text{V} \\ \text{AGR} & \begin{bmatrix} \text{NUM} & \text{pl} \end{bmatrix} \end{bmatrix} \qquad \textit{screech} \rightarrow \begin{bmatrix} \text{CAT} & \text{V} \\ \text{AGR} & \begin{bmatrix} \text{NUM} & \text{sg} \\ \text{PER} & \neg 3 \end{bmatrix} \end{bmatrix}$$

The lexicon's feature structures can be empty. A minimal lexicon entry therefore associates a terminal with a set containing a single empty feature structure.

5.2.2 Grammar Rules

We extend the grammar rules by adding target-side constraints. A rule takes the form

$$\text{C} \rightarrow \langle \gamma, \alpha, \sim, I \rangle$$

where C is a target non-terminal, γ is a string of source terminals and non-terminals, α is a string of target terminals and non-terminals, $\sim$ is a one-to-one correspondence between source and target non-terminals, and I is a set of constraints.

Our constraints are similar to those of PATR-II and other monolingual formalisms, but our formalism is not an exact generalisation of PATR-II to the synchronous case, the main difference being that our rules are always reducible to SCFG rules by discarding the constraints. As we saw in Section 4.3, the combinatory rules in PATR-II are an abstracted form of CFG rule (with constraints) that only by convention employ a CAT feature to specify the phrase-structure labels of constituents. Our rules are strictly SCFG rules augmented with a set of constraints. In this regard, our formalism is like LFG in that it requires a context-free backbone.

Our terminals and non-terminals are atomic symbols, but during derivation each target-side symbol is paired with a feature structure value. We refer to the value associated with a rule's head, C, as F_0. We refer to the values associated with the right-hand-side target symbols as $F_1, F_2, \ldots F_n$, where n is the number of symbols in the target right-hand-side α.

The constraints are identities involving feature structure values. There are three types of constraint: relative, absolute, and probabilistic.

Relative Constraints

A relative constraint is an identity between two feature structure values $F_i(\pi)$ and $F_j(\rho)$, where F_i and F_j are the values associated with the i-th and j-th target symbols of a rule, and π and ρ are feature paths referring to values within F_i and F_j. For example, the rule

$$\text{S} \rightarrow \text{X}_1\ \textit{schreien} \mid \text{NP}_1\ \textit{screech}$$

might have a constraint $F_1(\langle \text{AGR} \rangle) = F_2(\langle \text{AGR} \rangle)$ that requires identity between the AGR values of F_1 and F_2, the feature structure values associated with the target non-terminal NP and the target terminal *screech*.

When the meaning is unambiguous, we will refer to a feature structure value F_i using the corresponding target-side symbol. For the rule just given we will write

$$\begin{gathered}\text{S} \rightarrow \text{X}_1\ \textit{schreien} \mid \text{NP}_1\ \textit{screech} \\ \langle \text{NP AGR} \rangle = \langle \textit{screech AGR} \rangle\end{gathered}$$

Absolute Constraints

An absolute constraint is an identity between two feature structure values $F_i(\pi)$ and G, where π is a feature path and G is a constant value. For example, the rule just given might have a further constraint $F_2(\langle \text{CAT} \rangle) = \text{V}$ that requires identity between the CAT value of F_2 and the atomic value V. We will write

$$\begin{gathered}\text{S} \rightarrow \text{X}_1\ \textit{schreien} \mid \text{NP}_1\ \textit{screech} \\ \langle \textit{screech CAT} \rangle = \text{V}\end{gathered}$$

Probabilistic Constraints

A probabilistic constraint is an identity between a feature structure value $F_i(\pi)$ and a random variable with an associated probability distribution. As in the other constraint

types, π is a feature path and F_i is the feature structure value associated with one of the target symbols.

For our example rule, the absolute constraint might be relaxed to a probabilistic constraint that allows $F_2(\langle \text{CAT} \rangle)$ to take on two possible values, V or N with respective probabilities 0.95 and 0.05.

We will write this as

$$\begin{aligned} &\text{S} \rightarrow \text{X}_1 \textit{ schreien} \mid \text{NP}_1 \textit{ screech} \\ &\quad \langle \textit{screech CAT} \rangle = x \qquad\qquad P(X = x) = \{\text{V} : 0.95, \text{N} : 0.05\} \end{aligned}$$

5.2.3 Derivations

Like in SCFG, a derivation begins with a pair of source and target start symbols, X and S, and ends with a pair of source and target sentences. A sequence of production steps links the initial state to the final state. Additionally to SCFG, every target symbol in the derivation is paired with a feature structure value. The target start symbol, S, is paired with an empty feature structure value.

Derivation States

For each state in a derivation, we will write the source and target sentential forms, separated by a bar, followed by the sequence of target feature structure values. The initial state is therefore written

$$\text{X}_1 \mid \text{S}_1 \qquad [\,]$$

The subscripts indicate the one-to-one correspondence between non-terminal symbols, just as in the notation used for rules. An intermediate state will include a mix of terminals and non-terminals, with at least one source-target non-terminal pair. For example,

$$\text{X}_1\ \text{X}_2 \textit{ schreien} \mid \text{DET}_1\ \text{N}_2 \textit{ screech} \qquad \left[\text{AGR } \boxed{1}\right] \quad \left[\text{AGR } \boxed{1}\right] \quad \left[\text{AGR } \boxed{1}\left[\text{NUM } \texttt{pl}\right]\right]$$

The three feature structures correspond to the DET, N, and *screech* target symbols, respectively. Co-index boxes indicate the sharing of values (re-entrancy) between feature structures within a sequence. In our example, all three feature structures share a common AGR value.

In the final state, the source and target sentential forms consist of terminals only.

Production Steps

Each production step links one state to the next by the application of some rule r. This involves three substeps: i) rewriting a linked pair of source and target non-terminal symbols with the source and target symbol sequences from the body of r, ii) rewriting a feature structure value with a sequence of values, one for each target side symbol in r's body, and iii) unifying the new feature structure values from step ii) according to r's constraints.

The first substep is performed just as in SCFG derivation. For the rewrite to occur, r's head non-terminal must match the target non-terminal that is to be rewritten. The second and third substeps are more complex, involving the selection of feature structure values from the lexicon and the satisfaction of constraints. As an illustration, we will consider the application of the rule

$$\begin{aligned} &\mathrm{S} \rightarrow \mathrm{X}_1 \; \textit{schreien} \mid \mathrm{NP}_1 \; \textit{screech} \\ &\quad \langle \mathrm{NP} \; \mathrm{AGR} \rangle = \langle \textit{screech} \; \mathrm{AGR} \rangle \\ &\quad \langle \textit{screech} \; \mathrm{CAT} \rangle = \mathtt{V} \end{aligned}$$

given the initial state

$$\mathrm{X}_1 \mid \mathrm{S}_1 \qquad []$$

The rule's head non-terminal, S, matches the non-terminal in the target sentential form and therefore the X-S non-terminal pair can be rewritten using the source and target symbol sequences from the rule's body:

$$\mathrm{X}_2 \; \textit{schreien} \mid \mathrm{NP}_2 \; \textit{screech} \qquad []$$

We use distinct indices for the newly-inserted non-terminals to distinguish them from non-terminals that were introduced in previous steps.

Having rewritten the non-terminal pair, we proceed to rewrite the feature structure that corresponds to the target non-terminal. We will refer to this feature structure value as F_0 and we refer to the values in the replacement sequence as $F_1, F_2, \ldots, F_n$. The initial value of an element F_i depends on whether the i-th target symbol in r's body is a terminal or a non-terminal: if it is a terminal, t, then a value is selected from the set of feature structure values in t's lexicon entry. If it is a non-terminal then F_i is initially empty. For example, if the lexicon entry for *screech* includes the feature structure

$$\begin{bmatrix} \mathrm{CAT} & \mathtt{V} \\ \mathrm{AGR} & \begin{bmatrix} \mathrm{NUM} & \mathtt{pl} \end{bmatrix} \end{bmatrix}$$

then we can rewrite the feature structure value as follows:

$$\mathrm{X}_2\ \textit{schreien}\ \mid\ \mathrm{NP}_2\ \textit{screech} \qquad [] \qquad \begin{bmatrix} \textsc{cat} & \mathtt{V} \\ \textsc{agr} & \begin{bmatrix} \textsc{num} & \mathtt{pl} \end{bmatrix} \end{bmatrix}$$

The final substep is to apply the rule's constraints. This is done by unifying the values on the two sides of each identity. The first constraint is $\langle \textsc{np}\ \textsc{agr} \rangle = \langle \textit{screech}\ \textsc{agr} \rangle$, which requires unification between the AGR values of F_1 and F_2. This results in F_2's AGR value being shared with F_1:

$$\mathrm{X}_2\ \textit{schreien}\ \mid\ \mathrm{NP}_2\ \textit{screech} \qquad \begin{bmatrix} \textsc{agr} & \boxed{1} \end{bmatrix} \qquad \begin{bmatrix} \textsc{cat} & \mathtt{V} \\ \textsc{agr} & \boxed{1} \begin{bmatrix} \textsc{num} & \mathtt{pl} \end{bmatrix} \end{bmatrix}$$

The second constraint is $\langle \textit{screech}\ \textsc{cat} \rangle = \mathtt{V}$, which requires unification between the CAT value of F_2 and the constant value V. Unification does not change the value of F_2 and so the state just shown is the result of this production step.

5.2.4 Example: Modelling Subject-Verb Agreement

Section 4.4 gave an example of a monolingual unification grammar fragment that modelled subject-verb agreement in English. We now give an example synchronous fragment, G_2, that generates German-English sentence pairs whilst imposing the same agreement constraints on the English side. The lexicon is unchanged from that of Figure 4.5 but we reproduce it here for convenience (Figure 5.1). The synchronous grammar rules are shown in Figure 5.2.

Figure 5.3 shows the derivation of a sentence pair using the grammar G_2. Figure 5.4 shows an incomplete derivation that starts out identically, but differs in the selection from the lexicon of a feature structure for *screech*. This choice leads to a dead-end.

Finally, Figure 5.5 shows the target-side derivation tree that results from the derivation in Figure 5.3.

$$
\begin{array}{lll}
\textit{the} \rightarrow \begin{bmatrix} \text{CAT} & \texttt{DET} \\ \text{AGR} & \begin{bmatrix} \text{PER} & 3 \end{bmatrix} \end{bmatrix} & \qquad & \textit{an} \rightarrow \begin{bmatrix} \text{CAT} & \texttt{DET} \\ \text{AGR} & \begin{bmatrix} \text{NUM} & \texttt{sg} \\ \text{PER} & 3 \end{bmatrix} \end{bmatrix} \\
\textit{owl} \rightarrow \begin{bmatrix} \text{CAT} & \texttt{N} \\ \text{AGR} & \begin{bmatrix} \text{NUM} & \texttt{sg} \\ \text{PER} & 3 \end{bmatrix} \end{bmatrix} & & \textit{owls} \rightarrow \begin{bmatrix} \text{CAT} & \texttt{N} \\ \text{AGR} & \begin{bmatrix} \text{NUM} & \texttt{pl} \\ \text{PER} & 3 \end{bmatrix} \end{bmatrix} \\
\textit{screeches} \rightarrow \begin{bmatrix} \text{CAT} & \texttt{V} \\ \text{AGR} & \begin{bmatrix} \text{NUM} & \texttt{sg} \\ \text{PER} & 3 \end{bmatrix} \end{bmatrix} & & \textit{screech} \rightarrow \begin{bmatrix} \text{CAT} & \texttt{V} \\ \text{AGR} & \begin{bmatrix} \text{NUM} & \texttt{sg} \\ \text{PER} & \neg 3 \end{bmatrix} \end{bmatrix} \\
\textit{screech} \rightarrow \begin{bmatrix} \text{CAT} & \texttt{V} \\ \text{AGR} & \begin{bmatrix} \text{NUM} & \texttt{pl} \end{bmatrix} \end{bmatrix} &
\end{array}
$$

Figure 5.1: Example grammar G_2 (lexicon)

$$
\begin{array}{ll}
\text{DET} \rightarrow \textit{die} \mid \textit{the} & (r_1) \\
\quad \langle \textit{the}\ \text{CAT} \rangle = \texttt{DET} & \\
\quad \langle \text{DET}\ \text{AGR} \rangle = \langle \textit{the}\ \text{AGR} \rangle & \\
\\
\text{V} \rightarrow \textit{schreien} \mid \textit{screech} & (r_3) \\
\quad \langle \textit{screech}\ \text{CAT} \rangle = \texttt{V} & \\
\quad \langle \text{V}\ \text{AGR} \rangle = \langle \textit{screech}\ \text{AGR} \rangle & \\
\\
\text{S} \rightarrow \text{X}_1\ \text{X}_2 \mid \text{NP}_1\ \text{V}_2 & (r_5) \\
\quad \langle \text{NP}\ \text{AGR} \rangle = \langle \text{V}\ \text{AGR} \rangle &
\end{array}
\qquad
\begin{array}{ll}
\text{N} \rightarrow \textit{Eulen} \mid \textit{owls} & (r_2) \\
\quad \langle \textit{owls}\ \text{CAT} \rangle = \texttt{N} & \\
\quad \langle \text{N}\ \text{AGR} \rangle = \langle \textit{owls}\ \text{AGR} \rangle & \\
\\
\text{NP} \rightarrow \text{X}_1\ \text{X}_2 \mid \text{DET}_1\ \text{N}_2 & (r_4) \\
\quad \langle \text{NP}\ \text{AGR} \rangle = \langle \text{DET}\ \text{AGR} \rangle & \\
\quad \langle \text{NP}\ \text{AGR} \rangle = \langle \text{N}\ \text{AGR} \rangle &
\end{array}
$$

Figure 5.2: Example grammar G_2 (rules)

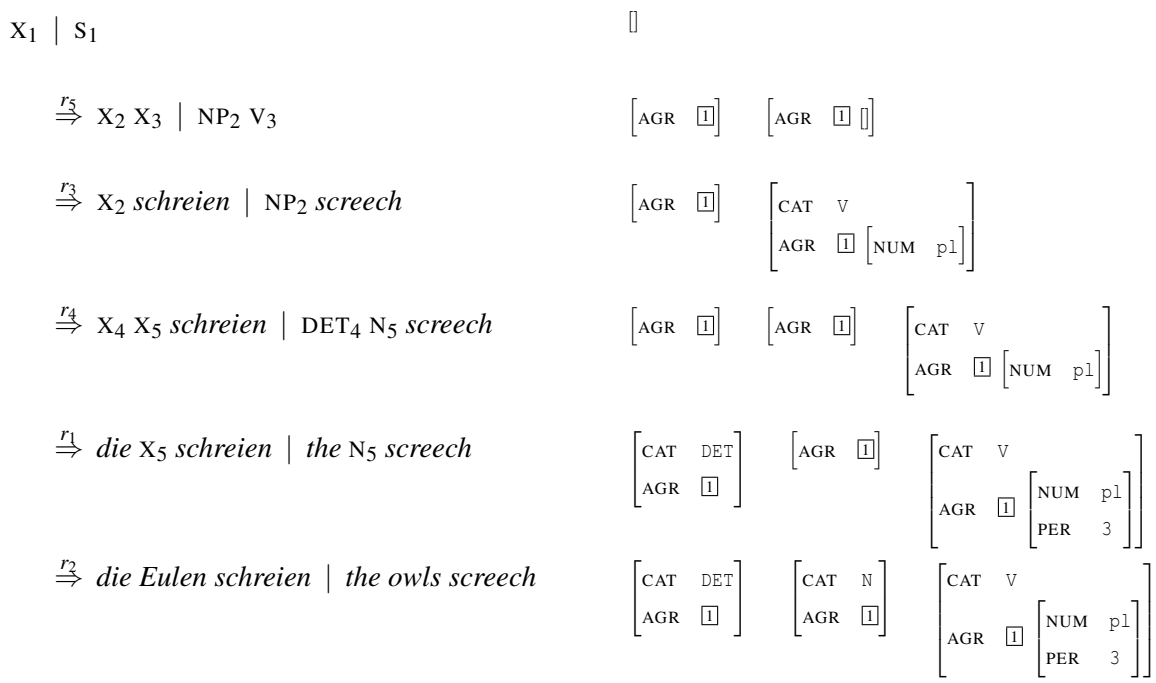


Figure 5.3: An example derivation of a sentence pair using the production rules and lexicon from grammar G_2. The feature structure sequences correspond to the target-side terminal / non-terminal sequences of each intermediate form.

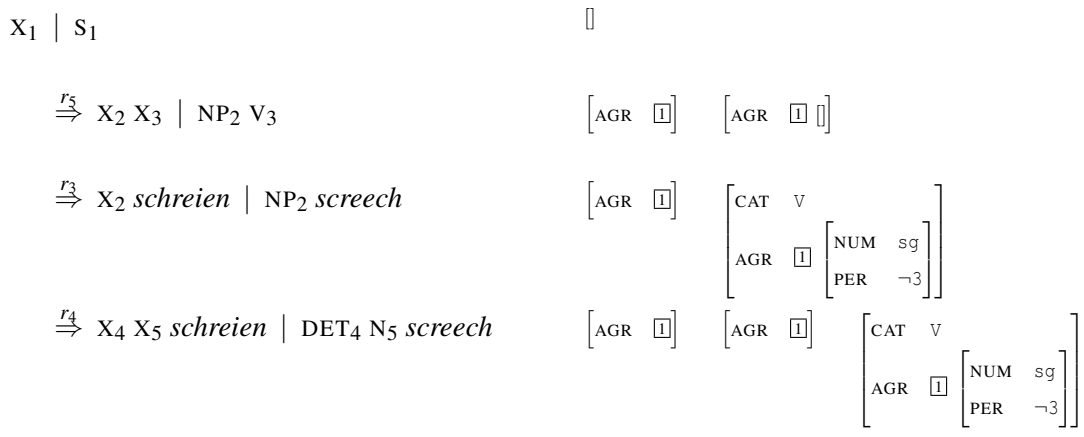


Figure 5.4: An incomplete derivation. The same production rules are applied as in Figure 5.3, but the second step differs in the choice of lexical entry for *screech*. This leads to a dead-end: only one further rule can be applied and no sentence pair can be derived.

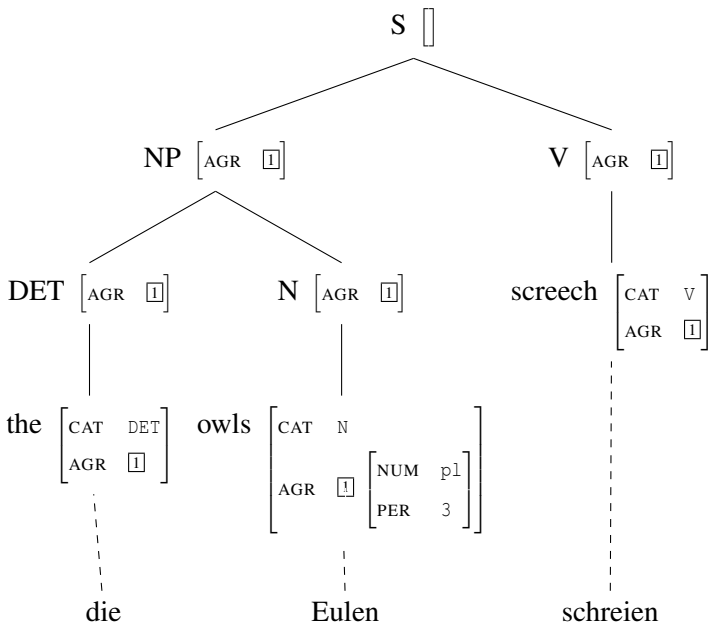


Figure 5.5: Target-side derivation tree corresponding to the derivation in Figure 5.3. The aligned source words are shown, but the source-side derivation tree is not.

5.3 Decoding

We saw in Chapter 2 that decoding is an approximate search for the highest-scoring translation t of a sentence s according to a model of $p(t|s)$. We saw in Chapter 3 that the search space of a string-to-tree (or similar syntax-based) model can be represented as a rooted, directed hypergraph and that decoding is then an approximate search for the optimal path through this hypergraph. By adding constraints to the grammar, we introduce a means of identifying ungrammatical paths in the search hypergraph.[1] Faced with a constraint failure, we can either remove the path from consideration (a hard constraint) or we can downweight it (a soft constraint). In this section, we briefly elaborate on the distinction between hard and soft constraints and then we describe the extended search hypergraph and how constraint evaluation is integrated into the decoding process.

[1]Technically, adding constraints also changes the structure of the hypergraph by splitting search states. We return to this point later.

5.3.1 Hard Versus Soft Constraints

Typically when parsing with a monolingual constraint-based grammar, the parser will reject the application of a rule in a context where the rule's constraints cannot be satisfied. If the parser assigns scores to derivations — as our string-to-tree translation decoder does — then an alternative is to permit the application but to incur a penalty so that the otherwise-illegal derivation is downweighted. The former approach uses *hard* constraints and the latter *soft* constraints.

Clearly, allowing soft constraints risks diminishing the capability of the constraint system to enforce grammaticality. However, soft constraints may be appropriate if the constraint extraction process is noisy or if the linguistic phenomenon that is being modelled is not completely regular and fully accounting for irregularity is difficult.

The choice of using hard or soft constraints will depend on the specific application and therefore our framework allows for both options. A natural means of defining a penalty for ill-formed derivations is to add a feature function that counts constraint failures. In Chapter 7 we compare the use of hard and soft constraints for one application.

5.3.2 The Expanded Search Hypergraph

In Section 3.3.5 we described the hypergraph structure of the string-to-tree search space. The label of each node encoded the category, the source span, and the m-gram language model context that were common to a set of subderivations. Each hyperedge was labelled with the synchronous rule that could be applied to the source subderivations to yield the target subderivations.

Adding constraints to the model requires that the hypergraph node labels also encode information about the subderivation's feature structure values. The minimal information that must be encoded is the single feature structure value for the root of the subderivation tree. In our description of the derivation process (Section 5.2.3), this was the F_0 value. In the context of bottom-up hypergraph search, we refer to this value as the *frontier feature structure value*. It is this value that determines whether a node can satisfy the constraints required for a traversal.

Figure 5.6 shows a very simple search hypergraph with hyperedges corresponding to applications of the rule r_3 from grammar G_2 (Figure 5.2) with the two different lexical feature structure values, f_1 and f_2, for the terminal *screech*.

Since the choice of lexical feature structures does not affect the score of a sub-

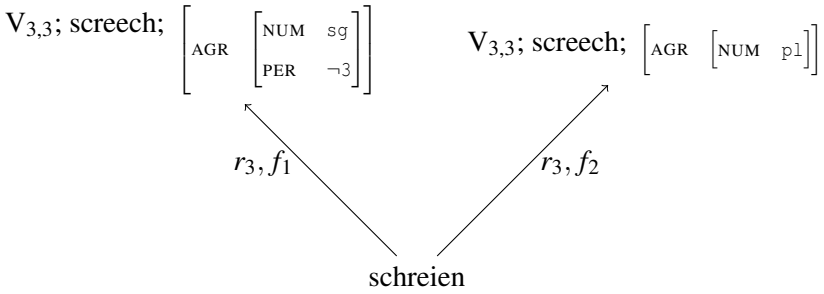


Figure 5.6: Search hypergraph in which the node labels distinguish states based on category, source span, m-gram LM context, and frontier feature structure value. On the hyperedge labels, f_1 and f_2 refer to the two lexical feature structure values for the target word *screech*.

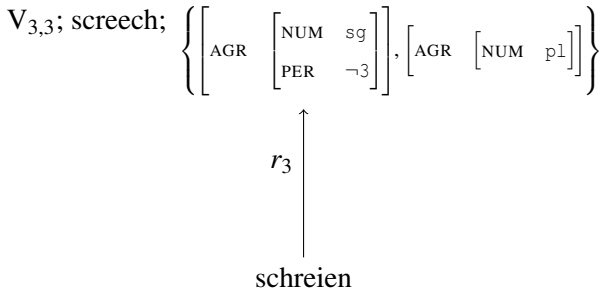


Figure 5.7: Search hypergraph in which the node labels distinguish states based on category, source span, m-gram LM context, and a set of frontier feature structure values.

derivation, we can collapse the states reached by the traversals r, f_1 and r, f_2 into a single state that records both possible frontier feature structure values. This is depicted in Figure 5.7.

These two styles of hypergraph can both be used to encode the same space of derivations, but the reduction in states in the latter can be important when using an approximate search strategy like cube pruning where only a subset of all possible states can be considered. We therefore adopt this latter style of hypergraph for decoding.

It is important to be clear aboout how constraint model state is defined. Recall that for an m-gram language model, state is defined by the boundary target words of a subderivation. If two subderivations yield the same boundary target words then their states are equivalent (with respect to the language model). For the constraint model, state is *partially* defined by the frontier feature structure value of a subderivation. State can only be fully defined by considering a set of related subderivations: specifically, it is defined as the set of frontier feature structure values from the set of subderivations that are identical other than for the choices of lexical feature structure values. Two constraint model states are equivalent if they contain identical sets of feature structure values.

To give a more complete example, Figure 5.8 shows the hypergraph that represents the search space for grammar G_2 when decoding the input sentence *die Eulen schreien*. We assume a bi-gram language model and therefore language model state records one left boundary word and one right boundary word. This hypergraph includes the sentential derivation from Figure 5.3. Note that the intermediate state reached by applying r_3 to *schreien* contains two feature structure values due to there being two lexical entries for *screech*. The presence of the second value allows the rule r_5 to be applied. With only the first value, this state would be a dead end. Note also that once the S node is reached, the feature structure values are no longer relevant (indicated by the empty feature structure value of the state).

The Feature Structure Sequence

Our search hypergraph does not record the details of which predecessor feature structures and which lexical feature structures participate in a traversal, not does it record non-frontier values. Usually, we are only interested in the target sentence strings that are produced as the end result of decoding and so this loss of derivational detail does not pose a problem. If the feature structure values were needed then the full derivation — or rather the set of derivations corresponding to the common SCFG derivation —

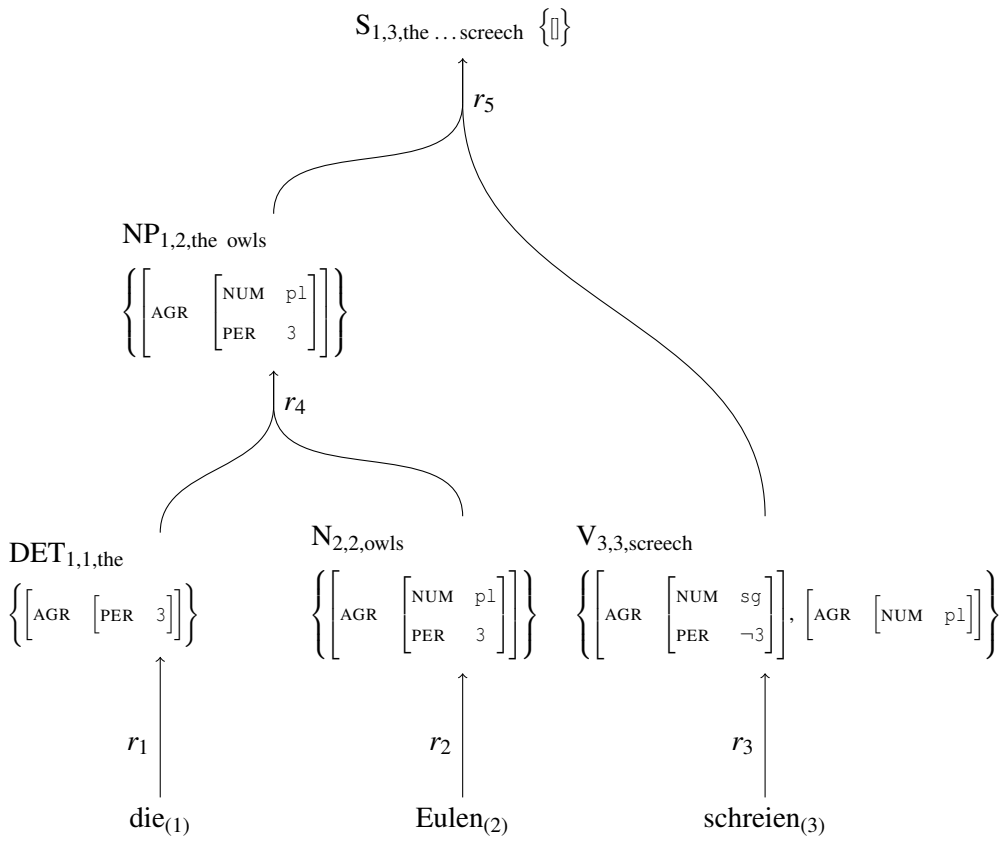


Figure 5.8: Search hypergraph for the input sentence *die Eulen schreien* and the grammar G_2.

could be reconstructed as a post-decoding step.

5.3.3 Integrating Constraint Evaluation

We integrate constraint evaluation into the decoding process in a similar manner to the m-gram language model. The requirement that the grammar has a context-free backbone means that constraint evaluation can be performed as a distinct secondary step of a rule application. We exploit this property for integration, leaving the decoder's context-free parsing algorithm (Section 3.3) unchanged; instead, the evaluation of a synchronous derivation's constraints is deferred until the beam-filling step (Section 3.3.6). We use cube pruning as the beam-filling strategy in experiments and in our description here, but we note that our method is compatible with other approximate search strategies such as that of Heafield et al. (2013).

Figure 5.9 shows the cube pruning algorithm adapted for hard constraints. Constraint evaluation is performed immediately prior to the point that a hypothesis is to

```
CUBE-PRUNE-HC(n, k)
 1  beam = empty ordered list
 2  q = empty priority queue
 3  for e in n's hyperedge set
 4      PUSH(q, CUBE-PRUNE-HYPEREDGE(e))
 5  while |beam| < k and |q| > 0
 6      lazy-list = POP(q)
 7      (hypothesis, score) = POP(lazy-list)
 8      if EVAL-CONSTRAINTS(hypothesis)
 9          ADD-TO-BEAM(beam, (hypothesis, score))
10      if lazy-list is not empty
11          PUSH(q, lazy-list)
12  return beam
```

Figure 5.9: The cube pruning algorithm modified to include hard constraints.

be added to the beam, the only change being that the call to ADD-TO-BEAM (line 9) is made conditional on the result of a call to EVAL-CONSTRAINTS, a function that returns true if the constraints can be satisifed for a hypothesis and false otherwise. The implementation of EVAL-CONSTRAINTS is described shortly. Note that for hard constraints, constraint failure does not affect the exploration of the cube since the failed hypotheses' neighbours will have been added to the priority queue as usual during the call to CUBE-PRUNE-HYPEREDGE.

When using soft constraints, a constraint failure can be reflected in the score. This requires no direct change to the cube pruning algorithm, only that the SCORE function that is called by CUBE-PRUNE-HYPEREDGE (Figure 3.10) calls a variant of EVAL-CONSTRAINTS and incorporates the result into the score calculation. Unlike for hard constraints, soft constraint failures can affect the exploration of the cube, since the score dictates the priority of items in the queue.

The EVAL-CONSTRAINTS Algorithm

For decoding, we are interested in a bottom-up exploration of the search graph that dynamically constructs nodes by applying rules to existing source nodes. Our adoption of set-based constraint model state means that a single traversal may involve taking into account multiple frontier feature structure values for each source node and multiple lexical feature structure values for each terminal of the rule. The EVAL-CONSTRAINTS algorithm is used to determine if any combination of incoming feature structures can

```
EVAL-CONSTRAINTS-NAIVE(h)
1  r = top-level rule in h
2  cs = r's constraint set
3  frontier-values = empty set
4  for fs-tuple in CARTESIAN-PRODUCT(sequence-of-incoming-fs-sets)
5       PREPEND(fs-tuple, [])
6       if SOLVE-CONSTRAINTS(fs-tuple, cs)
7            INSERT(frontier-values, fs-tuple[0])
8  SET-FRONTIER-VALUES(h, frontier-values)
9  return |frontier-values| > 0
```

Figure 5.10: The naive constraint evaluation algorithm.

satisfy the constraints of the rule, and for any such combinations, what the resulting frontier feature structure value are.

We will present two versions of the EVAL-CONSTRAINTS algorithm. The first is simple, but inefficient. In order to illustrate how our second version improves on the first, we introduce a new example application using constraints. The application is somewhat artificial but serves to illustrate why care must be taken to avoid inefficiencies arising from a combinatorial explosion of potential analyses. In the next chapter, when we look at agreement and government in German, we will see some examples that occur in natural language.

The Naive Algorithm

Figure 5.10 shows a naive constraint evaluation algorithm. For a single hyperedge in the search graph, it determines the set of feature structures in the head node's frontier set based on the contents of the frontier sets of incoming nodes and on the constraints associated with the hyperedge's grammar rule. At line 4, it enumerates all combinations of incoming feature structures by generating the Cartesian product from the sequence of incoming feature structure sets. The feature structure prepended at line 5 corresponds to the head rule symbol. Since search operates in bottom-up order, this value is empty prior to constraint satisfaction. For each resulting feature structure tuple the algorithm then attempts to unify values according to the constraints (line 6). If unification succeeds for all constraints then the head value is added to the set of frontier values. At line 8, the frontier set is recorded for the hypothesis, meaning that it can be used for recombination when the hypothesis is added to the beam.

Example: Repeated Vowel Sounds

Suppose we wish our grammar to generate target sentences in which the subject and the verb share at least one vowel sound. For example, the target sentence *the kittens hiss* would be grammatical because the subject and verb both contain an ɪ sound[2], but *an owl screeches* would not. We could model this by using the lexicon to list the vowel sounds of nominals and finite verbs and using constraints to test for matching vowel pairs. For example, a German-English grammar that could generate the target sentence *the kittens skittered down the garden* might include these lexical entries:

$$\textit{the} \rightarrow \begin{bmatrix}\text{CAT} & \text{D}\end{bmatrix} \qquad \textit{down} \rightarrow \begin{bmatrix}\text{CAT} & \text{PREP}\end{bmatrix}$$

$$\textit{kittens} \rightarrow \begin{bmatrix}\text{CAT} & \text{N} \\ \text{VOWEL} & \text{ɪ}\end{bmatrix} \qquad \textit{kittens} \rightarrow \begin{bmatrix}\text{CAT} & \text{N} \\ \text{VOWEL} & \text{ə}\end{bmatrix}$$

$$\textit{skittered} \rightarrow \begin{bmatrix}\text{CAT} & \text{V} \\ \text{VOWEL} & \text{ɪ}\end{bmatrix} \qquad \textit{skittered} \rightarrow \begin{bmatrix}\text{CAT} & \text{V} \\ \text{VOWEL} & \text{ə}\end{bmatrix}$$

$$\textit{garden} \rightarrow \begin{bmatrix}\text{CAT} & \text{N} \\ \text{VOWEL} & \text{ɑː}\end{bmatrix} \qquad \textit{garden} \rightarrow \begin{bmatrix}\text{CAT} & \text{N} \\ \text{VOWEL} & \text{ə}\end{bmatrix}$$

and these rules:

r_1 : $\text{N} \rightarrow \textit{Kätzchen} \mid \textit{kittens}$
$\langle \text{N VOWEL} \rangle = \langle \textit{kittens} \text{ VOWEL} \rangle$
$\langle \textit{kittens} \text{ CAT} \rangle = \text{N}$

r_2 : $\text{NP} \rightarrow \textit{den Garten} \mid \textit{the garden}$
$\langle \text{NP VOWEL} \rangle = \langle \textit{garden} \text{ VOWEL} \rangle$
$\langle \textit{garden} \text{ CAT} \rangle = \text{N}$

r_3 : $\text{NP} \rightarrow \textit{die } \text{X}_1 \mid \textit{the } \text{N}_1$
$\langle \text{NP VOWEL} \rangle = \langle \text{N VOWEL} \rangle$

r_4 : $\text{S} \rightarrow \text{X}_1 \textit{ liefen } \text{X}_2 \textit{ hinunter} \mid$
$\text{NP}_1 \textit{ skittered down } \text{NP}_2$
$\langle \text{NP}_1 \text{ VOWEL} \rangle = \langle \textit{skittered} \text{ VOWEL} \rangle$
$\langle \textit{skittered} \text{ CAT} \rangle = \text{V}$

Prior to the application of rule r_4, the search hypergraph would contain the nodes shown in Figure 5.11 (assuming a bi-gram language model). Now consider what the EVAL-CONSTRAINTS-NAIVE algorithm does for a hypothesis formed by the application of the rule r_4. The incoming feature structure sets are the sets S_1, S_2, S_3, S_4 where:

- S_1 is the frontier set for the predecessor NP node that covers *die Kätzchen*
- S_2 is the set containing the two lexical feature structures for *skittered*
- S_3 is the set containing the one lexical feature structure for *down*
- S_4 is the frontier set for the predecessor NP node that covers *den Garten*

[2] In examples, we use the phonetic symbols from the International Phonetic Alphabet (IPA).

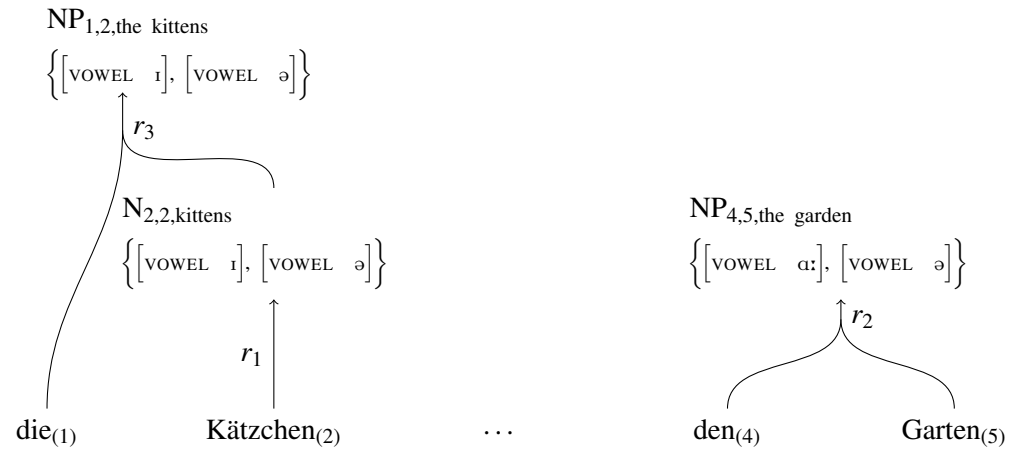


Figure 5.11: Partial search hypergraph for the input sentence *die Kätzchen liefen den Garten hinunter* and the grammar G_3.

At line 4, the algorithm enumerates all $|S_1|.|S_2|.|S_3|.|S_4| = 2 \times 2 \times 1 \times 2 = 8$ combinations of the incoming feature structures, evaluating constraints for each. This is wasteful because: i) the values in $|S_4|$ cannot affect the set of frontier values that is produced since r_4 has no constraints involving NP_2, and ii) there are two combinations involving ɪ for the subject's vowel and ə for the verb's (and vice versa), both of which fail unification, regardless of any other values. If a pair (or some larger subset) of incoming feature structures is found to be non-unifiable in one combination then no other combination containing that pair (or subset) need be enumerated.

The Improved Algorithm

Our second version of EVAL-CONSTRAINTS improves on the first in the following ways:

1. Feature structure tuples are built up progressively, starting with the elements of the single incoming feature structure set S_1. These initial tuples are extended by adding elements from S_2 to form tuples from the Cartesian product $S_1 \times S_2$, then tuples from $S_1 \times S_2 \times S_3$, etc. At each step, the constraints are evaluated and tuples for which evaluation fails are eliminated. In the example, the partial tuples involving the non-unifiable subject and verb vowel choices ɪ and ɑː would be eliminated after the first expansion.

2. If the i-th target rule symbol does not occur in any of the rule's constraints then the incoming feature structure set S_i is not included in the Cartesian product. In

our example, only the incoming feature structure sets S_1 (for the subject NP) and S_2 (for the verb) are included: we include tuples from $S_1 \times S_2$ only, not tuples from $S_1 \times S_2 \times S_3 \times S_4$.

3. The constraint set is partitioned to form the minimal subsets in which no rule symbol is referenced by constraints from more than one subset. Consider a rule with the following target right-hand side,

   $$\text{NP}_1 \textit{ skittered , } \text{NP}_2 \textit{ bounded , and } \text{NP}_3 \textit{ meandered}$$

   The constraint set would be partitioned into three subsets, one for each subject-verb pair. Evaluation would be performed separately for each subset, reducing the maximum number of feature structure tuples from $O(|S_1|.|S_2|.|S_4|.|S_5|.|S_8|.|S_9)$ to $O(|S_1|.|S_2| + |S_4|.|S_5| + |S_8|.|S_9)$.

Figure 5.12 shows the second version of the constraint evaluation algorithm. It partitions the constraint set (line 3) and then calls the subroutine EVAL-CONSTRAINT-SUBSET for each subset. The index sets are the sets of rule symbol indices after partitioning. In the example just given, the three index sets are $\{1,2\}$, $\{4,5\}$, and $\{8,9\}$. The partition of a given constraint set does not depend on any external factors and so can be performed as a preprocessing step.

EVAL-CONSTRAINT-SUBSET generates tuples of feature structures, starting with the elements of the first incoming feature structure set (lines 3 and 4) and then iteratively extending the tuples (lines 5-14), subject to successful constraint evaluation (lines 10 and 11). If the index set includes the head rule symbol then the set of frontier values is recorded (lines 15-19) for the hypothesis.

5.4 Conclusion

In this chapter we presented the unification-based framework that is used throughout the rest of the thesis. The framework can be used to add constraints to string-to-tree models without requiring changes in the base model.

In the next we will describe our baseline system and then in the following three chapters we will describe applications of our framework to several prominent morphosyntactic problems in English-German translation. Some of the technical choices relating to constraint evaluation and search were informed by these morphosyntactic applications. The details become much more relevant in the context of translating into

```
EVAL-CONSTRAINTS(h)
1  r = top-level rule in h
2  cs = r's constraint set
3  for (subset, index-set) in PARTITION(cs)
4      if !EVAL-CONSTRAINT-SUBSET(h, subset, index-set)
5          return false
6  return true

EVAL-CONSTRAINT-SUBSET(h, constraint-subset, index-set)
 1  fs-tuples = empty list
 2  i = index-set[0]
 3  for fs in S_i
 4      INSERT(fs-tuples, INIT-TUPLE(fs))
 5  for i in index-set[1..|index-set| − 1]
 6      extended-tuples = empty list
 7      for fs in S_i
 8          for tuple in fs-tuples
 9              extended-tuple = EXTEND-TUPLE(tuple,fs)
10              if UNIFY(extended-tuple, constraint-subset)
11                  INSERT(extended-tuples, extended-tuple)
12      if |extended-tuples| = 0
13          return false
14      fs-tuples = extended-tuples
15  if 0 ∈ index-set
16      frontier-values = empty list
17      for tuple in fs-tuples
18          INSERT(frontier-values, tuple[0])
19      SET-FRONTIER-VALUES(h, frontier-values)
20  return true
```

Figure 5.12: The improved constraint evaluation algorithm.

richly morphological languages and in the following chapters we will see examples of feature value ambiguity that arise in natural language.

Chapter 6

Baseline Setup

6.1 Introduction

In this chapter we describe the baseline English-German system that is used in experiments throughout the following chapters. We first give an outline of the system before describing the data, rule extraction, and feature functions in more detail.

Our baseline closely resembles the string-to-tree system described in Williams and Koehn (2012), which was one of Edinburgh's submissions to WMT 2012's translation task and which performed competitively: based on human evaluation, it was ranked second highest out of the eight systems that were trained on the same data (Callison-Burch et al., 2012); based on BLEU, it was ranked joint third of nine systems. In the 2013 translation task, an identically configured system trained using the 2013 training data was the highest ranked of 11 systems, based on human evaluation (Bojar et al., 2013).

6.2 System Description

6.2.1 System Outline

Our baseline system uses a string-to-tree SCFG translation grammar and a 5-gram language model. We used the Moses SMT toolkit (Koehn et al., 2007b) for training the model and decoding, with some additional toolkits for subtasks.

The translation grammar was learned from a word-aligned English-German parallel corpus with phrase-structure parse trees on the target side. The corpus was automatically word-aligned using MGIZA++ (Gao and Vogel, 2008), a multi-threaded

implementation of GIZA++ (Och and Ney, 2003). The German-side of the parallel corpus was then parsed using the BitPar[1] parser. If a parse failed then the sentence pair was discarded. The SCFG grammar was then extracted using the GHKM algorithm, subject to scope-3 pruning.

The monolingual German data was used to train seven 5-gram language models (one each for Europarl, News Commentary, and the five News data sets). These were interpolated using weights optimised against the development set and the resulting language model was used in experiments. We used the SRILM toolkit (Stolcke, 2002) with Kneser-Ney smoothing (Chen and Goodman, 1998).

The feature function weights were tuned on the *news-test2008* dev set using the Moses implementation of minimum error rate training (Och, 2003).

For evaluation we use case-sensitive BLEU-4 (Papineni et al., 2002) with a single reference.

6.2.2 Data

Our systems use all of the available English-German data from the 2012 Workshop on Machine Translation (Callison-Burch et al., 2012). The parallel corpus, which is used to learn the translation grammar, is derived from two genres, European Parliamentary proceedings (92.4%) and news (7.6%). The monolingual corpus, which is used to learn the *m*-gram language model, is derived from the same two genres but in almost the opposite proportion: 6.6% European Parliamentary proceedings and 93.4% news.

The tuning set (newstest2008) and three test sets (newstest2009, newstest2010, newstest2011) are all drawn from the news genre.

Table 6.1 shows the sizes of the data sets used for training, tuning, and evaluation. Two values are shown for the parallel training corpus: the larger value is for the pre-processed corpus,[2] which is used for automatic word alignment. The parsed version, which is used for grammar extraction, is slightly smaller because BitPar is unable to parse some of the target sentences. These sentence pairs (0.7%) are discarded prior to grammar extraction.

The tuning and test sets all have a single reference translation.

[1] `http://www.ims.uni-stuttgart.de/tcl/SOFTWARE/BitPar.html`

[2] We use the standard Moses corpus filtering script, which removes sentence pairs where one or both sentences is overly long (80 tokens in this case) or where the pair has a dubious length ratio. Prior to cleaning the corpus size is 2.08M sentence pairs.

Data Set	Sentences	Data Set	Sentences
Parallel (preprocessed)	2.04M	Tuning	2,051
Parallel (parsed)	2.03M	Test (2009)	2,525
Monolingual	33.02M	Test (2010)	2,489
		Test (2011)	3,003

Table 6.1: Sizes of the training, tuning, and evaluation data sets.

6.2.3 Rule Extraction

The SCFG translation grammar is extracted from the word-aligned parallel training data using the Moses implementation (Williams and Koehn, 2012) of the GHKM algorithm that was described in Section 3.2.2. Minimal rules are composed into larger rules subject to the following limitations, defined in terms of the target tree fragment:

Rule depth the maximum distance from the composed rule's root node to any other node within the fragment, not counting preterminal expansions. This is set to 3.

Node count the number of target tree nodes in the composed rule, excluding target words. This is set to 15.

Rule size the measure defined in DeNeefe et al. (2007): the number of non-part-of-speech, non-leaf constituent labels in the target tree. This is set to 3.

Fully non-lexical unary rules are eliminated using the method described in Chung et al. (2011a). Rules with scope greater than 3 (Section 3.3.3) are not added to the translation grammar.

6.2.4 Feature Functions

Our feature functions assign various scores to the synchronous derivations that are produced during translation. As explained in Section 3.3, the m-gram language model probability of the derivation's target yield cannot be computed by summing scores for the individual rules. However, the remaining scores *can* be decomposed in this way. Each grammar rule is therefore associated with a set of pre-computed scores, one for each feature function. For a grammar rule of the form

$$C \rightarrow \langle \gamma, \alpha, \sim \rangle$$

where C is a target-side non-terminal label, γ is a string of source terminals and non-terminals, α is a string of target terminals and non-terminals, and $\sim$ is a one-to-one correspondence between source and target non-terminals, we score the rule according to the following functions:

- $p(\gamma \mid C, \alpha)$, the noisy-channel translation probability.
- $p(C, \alpha \mid \gamma)$, the direct translation probability.
- $p_{lex}(\alpha \mid \gamma)$ and $p_{lex}(\gamma \mid \alpha)$, the direct and indirect lexical weights (Koehn et al., 2003).
- $p_{pcfg}(\pi)$, the monolingual PCFG probability of the tree fragment π from which the rule was extracted. This is defined as $\prod_{i=1}^{n} p(r_i)$, where $r_1 \ldots r_n$ are the constituent CFG rules of the fragment. The PCFG parameters are estimated from the parse of the target-side training data. All lexical CFG rules are given the probability 1. This is similar to the p_{cfg} feature used in Marcu et al. (2006) and is intended to encourage the production of syntactically well-formed derivations.
- $\exp(-1/count(r))$, a rule rareness penalty.
- $\exp(1)$, a rule penalty.

6.2.5 Glue Rules

During development we found that allowing MERT to learn a glue rule penalty led to highly variable levels of glue rule use between optimisation runs,[3] suggesting that the tuning metric, BLEU, is not particularly sensitive to higher-level syntactic structure. Fixing the glue rule penalty at a strongly negative weight (we used -1) forces the generation of syntactic structure and we found that this had little effect on the final BLEU score. See Table 6.2 for average scores on the three test sets. Following the recommendation of Clark et al. (2011), we ran the MERT optimization step three times for each system and repeated the evaluation with each set of feature weights.

Given the size of the translation grammar and given that GHKM grammars include highly-permissive non-lexical rules, it is perhaps not too surprising that when forced not to use glue rules, the decoder is able to find some combination of high-level rules

[3] Although in the final experiments presented here, glue rule use happened to be very consistent over MERT runs at an average of 2.4 glue rules per sentence with a standard deviation of only 0.02.

Test Set	2009		2010		2011	
Glue	15.0	0.0	16.5	0.1	15.3	0.1
No glue	15.1	0.1	16.6	0.1	15.4	0.2

Table 6.2: Average BLEU scores and standard deviations for the three test sets, with and without glue rules.

that are no worse than glue rules, especially if the preference that the decoder might otherwise have had for using glue rules was based on weak evidence in the first place.

6.3 Conclusion

In this chapter we presented the baseline system that is used throughout the rest of the thesis. A similar system was found to perform strongly in the WMT 2012 translation task.

In the next three chapters we will describe applications of our framework to several prominent morphosyntactic problems in English-German translation. Namely, the issues of agreement, government, and discontiguous verb complex construction.

Chapter 7

Agreement and Government

If you want to do agreement, I don't think the right way of doing it is to look back at 5-grams, you just need so many of them. Better to look back and ask questions about, where's the noun? — Peter Brown (2013)

7.1 Introduction

In this chapter we apply our framework to the task of modelling agreement and government in German. The majority of surface-form inflection in German and other Indo-European languages can be understood in terms of these phenomena and by modelling the underlying grammatical relationships we aim to improve the generation of inflectional surface forms in machine translation.

We first introduce the relevant linguistic concepts and explain why these phenomena pose a problem for SMT. We then look at the previous approaches that have been taken to modelling inflection in SMT before describing how to model German agreement and government phenomena using the feature structures and constraints of our framework. To evaluate the effectiveness of the approach, we extend the common baseline model and data that was described in Chapter 6 and then present experimental results and analysis.

An early version of this work was presented in Williams and Koehn (2011). In that work we found soft constraints to be more effective than hard constraints, whereas we now find the opposite. Since we believe this to be due to improvements in our training process, we provide a summary of the main differences in the training compared to Williams and Koehn (2011). This chapter also includes extensions and additional experiments that were developed subsequently.

7.2 Background

7.2.1 Agreement and Government

In natural language, *agreement* is the repeated marking of grammatical features in syntactically distinct parts of a sentence. We have already seen the example of subject-verb agreement in English, where the number of a common noun is usually expressed via inflection on the noun itself (*owl* versus *owls*) and if the noun is the subject of a clause then the same information is re-expressed on the finite verb of the clause (*screeches* versus *screech*). Agreement relations are asymmetric in the sense that one participant is the source of the information, sometimes called the *controller*, and the rest are targets. In subject-verb agreement, the subject noun or pronoun is the source of the information and the verb is the target.

Cross-linguistically, the main grammatical features involved in agreement are gender, number, and person. It is typical that the controller is nominal in nature (Corbett, 2006, p7).

Government is closely related to agreement. Like in agreement, a source element (the *governer*) determines one or more features that are marked on syntactically distinct parts of the sentence. However, unlike agreement, the features are not marked on the source element itself. An example is verbal case government in Russian, where the lexical choice of the verb determines the cases of the noun phrase complements.

We will use the more general term *selector* to refer to both controllers and governers.

7.2.2 Inflection

Agreement and government figure prominently in morphologically-rich languages. Although their broad definitions cover other possibilities (such as clitic agreement), agreement and government usually involve the coordination of inflectional affixes.

Whilst English makes limited use of inflectional morphology, many languages use inflection to mark a much wider range of grammatical distinctions. Corbett (2012, p74) gives the following examples[1] from Russian showing gender marking on nouns and verbs:

[1] In examples, we use the Leipzig Glossing Rules: `http://www.eva.mpg.de/lingua/resources/glossing-rules.php`

(1) Žurnal by-l zdes′
magazine(M).SG.NOM be-PST[M] here
'The magazine was here'

(2) Kniga by-l-a zdes′
book(F).SG.NOM be-PST-F here
'The book was here'

As in English, the verb is inflected to agree with the subject, but in Russian this includes gender agreement: *byl* agrees with the masculine noun *Žurnal* and *byla* agrees with the feminine noun *Kniga*.

Cross-linguistically, grammatical features can be marked for word classes that are uninflected in English, such as determiners and adjectives in Russian. For non-nominal word classes, the inflection is usually defined by one or more agreement or government relations.

7.2.3 Inflection and SMT

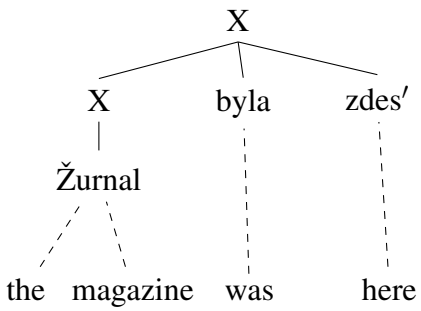
The coordination of inflectional markers poses a problem for statistical machine translation since the words that bear the markers may be produced by the application of independent translation rules. Typically, the *m*-gram language model is the only means of enforcing consistency.

Consider the following hierarchical SCFG rules, which could be learned from a corpus containing the two English-Russian translations from Section 7.2.2:

$\text{X} \rightarrow \textit{the magazine} \mid \textit{Žurnal}$ $\quad$ $\text{X} \rightarrow \text{X}_1 \textit{ was here} \mid \text{X}_1 \textit{ byl zdes′}$

$\text{X} \rightarrow \textit{the book} \mid \textit{Kniga}$ $\quad$ $\text{X} \rightarrow \text{X}_1 \textit{ was here} \mid \text{X}_1 \textit{ byla zdes′}$

These rules can produce correct translations of the two English sentences, but they also allow ungrammatical derivations such as:



Producing correctly inflected translations is particularly challenging for translation into languages with rich morphology where features, like gender in Russian, are often not present in the source language.

7.3 Previous Work

There is a large body of work that addresses the problem of producing accurate target morphology, most of which is motivated by similar problems of agreement and government. The majority of this work has been based on phrase-based models.

7.3.1 Inflection Marking as Post-Processing

To improve translation in language pairs with complex source-side inflection, an effective approach has been to simplify the source-side of the data, either by stemming or a similar word-clustering approach (Nießen and Ney, 2001; Goldwater and McClosky, 2005; Talbot and Osborne, 2006). Much of the information encoded in the morphological distinctions is redundant for translation (for example, source-side gender distinctions when translating into a language without grammatical gender) and simplifying the data has the advantage of reducing data sparsity for word alignment and in the translation model. For rich target-side side morphology, a similar approach can be taken, provided that there is also some means for restoring morphology.

Minkov et al. (2007) explore how a post-processing step might generate target inflection from stemmed translation output. They develop a maximum entropy model that predicts the target inflection given features from a small window of neighbouring words and aligned source words. The model is evaluated by measuring its accuracy at restoring the surface-forms of stemmed English-Arabic and English-Russian sentence pairs, where it significantly outperforms the random and language model baselines.

Toutanova et al. (2008) continue this line of research, applying Minkov et al.'s (2007) model directly to English-Arabic and English-Russian machine translation systems. They compare systems that re-inflect fully inflected output with those that inflect stemmed output, where the target input is either stemmed before or after word alignment. For their English-Russian dependency tree-based system, they find that all three methods improve BLEU score significantly, with stemmed word alignment and re-inflection providing similar and complementary improvements. For a phrase-based English-Russian system they only try re-inflection, finding a smaller improvement. An English-Arabic dependency tree-based system shows gains for re-inflection, with smaller gains for the methods that use stemmed training data. The authors suggest this is likely due to the loss of inflectional information that could otherwise be inferred from evidence in the English source.

Fraser et al. (2012) also proposes a two-step approach in which English is first translated into a morphologically-simplified form of German using a conventional phrase-based model and then the output is fully inflected using a CRF-based model. The intermediate German representation is produced by stemming the surface forms and then adding inflection-related tags according to a scheme that takes into account whether features are inherent or contextual for a given word class (for instance, gender tags are added to nouns, which are controllers, but not to adjectives, which are targets).

7.3.2 Factored Translation Models

In factored translation models (Koehn and Hoang, 2007), the surface-form tokens of phrase-based models are replaced with vectors of factors, where a factor is an arbitrary token. Typically, factors are used to represent linguistic types such as lemmas, part-of-speech tags, or morphological features. The translation process involves one or more factor mapping steps, each operating on a subset of source and target factors, followed by zero or more generation steps, which combine target factors to produce a final output form. Koehn and Hoang (2007)'s framework generalizes a number of lemma-tag approaches that were used in earlier work on morphology in SMT.

There is a large body of experimental work that applies factored models to translation issues in a diverse range of language pairs (Koehn et al., 2007a; Holmqvist et al., 2007; Bojar, 2007; Stymne et al., 2008; Avramidis and Koehn, 2008; Ramanathan et al., 2009). The most successful approaches are typically those where the number of mapping steps is kept small and any generation steps are carefully chosen such that the number of possible expansions is limited. Koehn and Hoang (2007) warn that the use of more complex factored models in their experiments was precluded by the combinatorial expansion of the search space associated with generating translation options and this is a danger in any system that generates surface forms from more general types. This is further cited as a factor in the decision by Toutanova et al. (2008) to construct an inflection model as an independent post-processing step rather than integrate it into the decoder.

7.3.3 Global Discriminative Models

In the log-linear models we saw in Chapter 2, the individual translation units — phrase pairs or synchronous rules — were scored independently of the context in which they were applied. For some aspects of translation, source context is crucial for accurate

unit selection and so researchers have explored approaches for integrating global discriminative models into SMT systems. For instance, Carpuat and Wu (2007) integrate a discriminative word sense disambiguation model that improves target word selection based on the full source context.

Jeong et al. (2010) and Subotin (2011) have both applied this approach to improving target morphology. Jeong et al. (2010) were able to improve translation quality for English into Bulgarian, Czech, and Korean by incorporating a discriminatively-trained log-linear model defined over large numbers of features involving various aspects of source and target morphology. Subotin (2011) took a similar approach for English-Czech, but also allowed for the generation of unseen Czech forms.

7.3.4 Morpheme Segmentation

For translation involving an agglutinative language, such as Finnish or Turkish, tokens are typically segmented into individual morphemes prior to translation. For English-Finnish translation, Clifton and Sarkar (2011) achieve an improvement in translation quality by using a phrase-based model defined over segmented tokens combined with a post-processing step similar to that of Toutanova et al. (2008).

Arabic-English translation typically also involves morphological segmentation (Lee, 2004; Zollmann et al., 2006). Although Arabic's morphological system is fusional rather than agglutinative, Arabic shares the characteristic that single multi-morpheme tokens frequently correspond to morphemes that would be distinct tokens in English. For translation *into* Arabic, Green and DeNero (2012) propose a model that involves both segmentation and agreement modelling. They represent agreement features as tags and train a CRF model to generate bi-gram tag sequences using features derived from the surface, such as prefixes, affixes, and indicators for digits. Segmentation and sequence tagging are both performed during decoding, as hypotheses are generated.

7.3.5 Unification-based Approaches

As discussed in Chapter 4, unification has been widely used in transfer-based approaches to machine translation. Agreement constraints are a natural component of the target-language translation step in these models and Lavie (2008) uses constraints to enforce number and gender agreement.

In a precursor to the current work, we developed a unification-based agreement checker within the hierarchical-phrase based framework (Williams, 2009). This was

limited to testing short sequences of words (strictly within the range of the m-gram language model) using pre-defined part-of-speech sequences to identify word sequences that were likely to share features. Whilst this approach was found to have a negligible effect on BLEU score, manual analysis revealed that the approach was successful at identifying agreement failures. It was observed that the decoder would frequently circumvent agreement constraints by producing syntactically ill-formed sequences, a finding which helped motivated the syntax-based approach proposed in this thesis.

7.3.6 Advantages of the Proposed Approach

Whilst most SMT research has historically been focussed on translation into English, there is now a diverse set of approaches for handling rich source or target morphology. Most of these approaches have been based on an underlying phrase-based model and have focussed on ensuring consistent inflection across phrasal boundaries. With the exception of the global discriminative models, the approaches have treated inflection as a sequence tagging problem, where the inflection of a word is informed by a window of neighbouring words. For example, Green and DeNero (2012) use a bi-gram agreement model for English-Arabic and Fraser et al. (2012) uses a CRF model for English-German with features defined over the preceeding and subsequent five words.

Our framework offers a number of advantages over existing approaches, in that it:

1. allows agreement and government relations to be defined in terms of target syntactic structure. For many target words, the relationship to the controller or governer is difficult to describe without reference to syntax.

2. places no inherent restriction on the range of agreement and government relations. Whilst many agreement issues are highly-localized, there are important exceptions that present difficulties for sequence models. For example, subject-verb agreement in languages with verb-final syntactic configurations.

3. produces inflected forms during search. The parameters for the model's feature functions can therefore be estimated from fully inflected forms. In practice, data sparsity may lead to worse rather than better parameter estimation, but our approach does not preclude the use of feature functions estimated from both inflected and uninflected data (although note that in the experiments presented here we use only inflected data).

7.4 Model

We apply our approach to German, which has a rich inflectional morphology and exhibits a range of agreement and government phenomena. This section describes how we model these phenomena using the lexicon and constraints of the formalism that was described in Chapter 5.

7.4.1 Internal NP and PP Inflection

Within a German noun phrase or prepositional phrase, inflectional suffixes are added to determiners, attributive adjectives, and nouns. The choice of suffix on an individual word is determined by multiple factors, which, with one exception, are internal to the phrase. The single external factor — case marking of noun phrases according to grammatical function — is deferred until Section 7.4.2.

The inflectional markers take the form of single-morpheme suffixes. These are highly *syncretic*: that is, a single suffix form may be shared between many feature values. Whilst the feature values of an inflected form may be ambiguous when the word is observed in isolation, they are usually unambiguous within the context of the phrase. For instance, in isolation the inflected adjective *großen* could be analysed as *big*-N.SG.ACC, as *big*-PL.GEN, or as one of nine other morphological analyses. In the noun phrase *der großen Hunde* the interpretation is unambiguously *the*.PL.GEN *big*.PL.GEN *dog*PL[GEN].

Noun-Modifier Agreement

Determiners and attributive adjectives are inflected to agree with the gender and number feature values that are inherent to the noun. Number is usually marked on the noun, as in English, but gender is not.

We model this in a similar manner to the English subject-verb agreement example in Section 5.2.4: the lexical entry for a noun includes agreement features that indicate the inherent gender and number values of the noun. The lexical entries for determiners and attributive adjectives contain agreement values that are consistent with their inflection.

As we will see shortly, it is convenient to treat the agreement value as one part of a larger structure that determines inflection. We refer to this larger feature value as the INFL feature.

Constraints are used to ensure that agreement values are compatible under unification. For example, given the following lexical entries:

$$\textit{das} \rightarrow \begin{bmatrix} \textsc{cat} & \texttt{ART} \\ \textsc{infl} & \begin{bmatrix} \textsc{agr} & \begin{bmatrix} \textsc{gender} & \texttt{n} \\ \textsc{num} & \texttt{sg} \end{bmatrix} \\ \ldots & \ldots \end{bmatrix} \end{bmatrix} \qquad \textit{Kätzchen} \rightarrow \begin{bmatrix} \textsc{cat} & \texttt{NN} \\ \textsc{infl} & \begin{bmatrix} \textsc{agr} & \begin{bmatrix} \textsc{gender} & \texttt{n} \\ \textsc{num} & \texttt{sg} \end{bmatrix} \\ \ldots & \ldots \end{bmatrix} \end{bmatrix}$$

the following rules would allow the translation of the input *the kitten* to the output *das Kätzchen*:

$$\begin{array}{ll} \textsc{art} \rightarrow \textit{the} \mid \textit{das} & \textsc{np-sb} \rightarrow \textsc{x}_1\ \textit{kitten} \mid \textsc{art}_1\ \textit{Kätzchen} \\ \quad \langle \textit{das}\ \textsc{cat} \rangle = \texttt{ART} & \quad \langle \textit{Kätzchen}\ \textsc{cat} \rangle = \texttt{NN} \\ \quad \langle \textsc{art}\ \textsc{infl} \rangle = \langle \textit{das}\ \textsc{infl} \rangle & \quad \langle \textsc{art}\ \textsc{infl} \rangle = \langle \textit{Kätzchen}\ \textsc{infl} \rangle \end{array}$$

Given constraints of this type, it is the absence of compatible lexical entries that prevents the production of grammatically incorrect derivations.

The relationship between the noun and modifiers is symmetrical in our model: no distinction is made between the controller and target of the agreement relation.

Prepositional Case Government

The case of a prepositional phrase is governed by the preposition. For example, a phrase headed by the preposition *mit* ('with') will always be in the dative case. Most prepositions either govern one case exclusively or govern two cases with the choice of case indicating a difference in meaning (usually relating to whether movement is involved).

We model this in a similar way to noun-modifier agreement: a lexical entry for a preposition contains a case value corresponding to a case governed by that preposition. For example, the preposition *unter* ('under') has two lexical entries, one for each of the cases it governs:

$$\textit{unter} \rightarrow \begin{bmatrix} \textsc{cat} & \texttt{APPR} \\ \textsc{infl} & \begin{bmatrix} \textsc{case} & \texttt{acc} \end{bmatrix} \end{bmatrix} \qquad \textit{unter} \rightarrow \begin{bmatrix} \textsc{cat} & \texttt{APPR} \\ \textsc{infl} & \begin{bmatrix} \textsc{case} & \texttt{dat} \end{bmatrix} \end{bmatrix}$$

The inflection of determiners, attributive adjectives, and nouns depends on the case of the noun phrase, and so their lexical entries also contain a case value that must be compatible under unification.

Adjectival Declension Type

In addition to nominal agreement and phrasal case, the choice of inflectional suffix on an attributive adjective is determined by one further factor: the adjectival declension paradigm. The choice of paradigm is determined by the presence or absence of a determiner within the phrase, and, if present, its definiteness. In the context of analysing German adjective agreement in GPSG, Zwicky (1986) argues that this phenomenon should be considered an instance of government.

The absence of a determiner requires the use of the most expressive adjectival inflection paradigm with five possible suffixes. This is referred to as 'strong' declension. If the phrase includes a definite article then the least expressive inflection paradigm is used, with two possible suffixes. This is referred to as 'weak' declension. For all other determiners, a hybrid inflection pattern is used — 'mixed' declension.

Like gender, number, and case, we model the choice of declension paradigm with a feature. The declension feature has an important difference that complicates modelling: the value depends not only on a property of the controller (the definiteness of a determiner), but also on the presence or absence of the controller. We therefore model declension control in two parts. The first is the same as for other features: determiners in the lexicon specify a declension value (mixed or weak) according to their definiteness and attributive adjectives have a declension value that matches their inflection. For example, lexical entries for the attributive adjective *wilde* ('fierce') and *Kätzchen* ('kitten' or 'kittens') are (among others):

$$\textit{wilde} \rightarrow \begin{bmatrix} \textsc{cat} & \texttt{ADJA} \\ \textsc{infl} & \begin{bmatrix} \textsc{case} & \texttt{nom} \\ \textsc{decl} & \texttt{strong} \\ \textsc{agr} & \begin{bmatrix} \textsc{num} & \texttt{pl} \end{bmatrix} \end{bmatrix} \end{bmatrix} \qquad \textit{Kätzchen} \rightarrow \begin{bmatrix} \textsc{cat} & \texttt{NN} \\ \textsc{infl} & \begin{bmatrix} \textsc{agr} & \begin{bmatrix} \textsc{num} & \texttt{pl} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Since noun inflection does not depend on the adjectival declension paradigm, lexical entries for nouns do not have a declension feature.

The second part to modelling is purely syntactic: if a determiner is absent from a phrase then a constraint is used to require that any attributive adjectives have the strong declension value. For example:

$$\begin{array}{l} \textsc{adja} \rightarrow \textit{fierce} \mid \textit{wilde} \\ \quad \langle \textit{wilde}\ \textsc{cat} \rangle = \texttt{ADJA} \\ \quad \langle \textsc{adja}\ \textsc{infl} \rangle = \langle \textit{wilde}\ \textsc{infl} \rangle \end{array} \qquad \begin{array}{l} \textsc{np-sb} \rightarrow \textsc{x}_1\ \textit{kittens} \mid \textsc{adja}_1\ \textit{Kätzchen} \\ \quad \langle \textit{Kätzchen}\ \textsc{cat} \rangle = \texttt{NN} \\ \quad \langle \textsc{adja}\ \textsc{infl} \rangle = \langle \textit{Kätzchen}\ \textsc{infl} \rangle \\ \quad \langle \textsc{adja}\ \textsc{infl}\ \textsc{decl} \rangle = \texttt{strong} \end{array}$$

7.4.2 NP Case Marking

The features and constraints in Section 7.4.1 are designed to enforce consistency of case marking within phrases. For prepositional phrases, the case is governed by the preposition, but for noun phrases we do not yet have any mechanism for making the overall case choice. As we have already mentioned, the primary use of case is to mark the grammatical relation of the noun phrase to the head verb — nominative case is used for subjects, accusative for the direct object, and so on. If we know the grammatical role of a noun phrase then we can constraint the case value of the phrase.

The Tiger corpus, on which the parser is trained, has several layers of annotation, including both phrase-structure and syntactic function labels (Brants et al., 2002). BitPar produces constituent labels that include both label types and our baseline system retains these. For example, the non-terminal label NP-SB is used for a noun phrase that functions as a subject. We thus have a ready source of syntactic annotation that pertains directly to case value choice.

We add constraints that set the case value for the noun phrase according to the grammatical function indicated by the constituent label. For example:

$$\begin{aligned}
&\text{NP-SB} \rightarrow \text{X}_1 \textit{ kittens} \mid \text{ADJA}_1 \textit{ Kätzchen} \\
&\quad \langle \textit{Kätzchen } \text{CAT} \rangle = \texttt{NN} \\
&\quad \langle \text{ADJA INFL} \rangle = \langle \textit{Kätzchen } \text{INFL} \rangle \\
&\quad \langle \text{ADJA INFL CASE} \rangle = \texttt{nom} \\
&\quad \langle \text{ADJA INFL DECL} \rangle = \texttt{strong}
\end{aligned}$$

Tiger uses a rich set of grammatical function labels and whilst the grammatical function label unambiguously determines the case value for the core role labels, it does not hold for all function labels. Therefore, for each of the function labels, we determine empirically from our training data whether to constrain the noun phrase case value or not. We will provide details in Section 7.5.2 once we have outlined the training process.

7.4.3 Subject-Verb Agreement

Like in English, finite verbs agree in number and person with their subjects. We model this exactly as we did for English in Section 5.2.4: the lexical entries for nouns include agreement features that indicate inherent number and person values; the lexical entries for finite verbs include number and person feature values that are consistent with the

Feature	Values	Target(s)				Determined By:
		Det	Adj	Noun	Verb	
case	nom, acc, gen, dat	✓	✓	✓		grammatical role of phrase, preposition
decl	strong, mixed, weak		✓			presence and definiteness of determiner
gender	m, f, n	✓	✓			inherent property of noun
number	sg, pl	✓	✓		✓	inherent property of noun
person	1, 2, 3				✓	inherent property of noun

Table 7.1: Lexical features and values used in this chapter.

inflection. We use the constituent labels to identify the participants in constraints. For example:

$$\begin{aligned} \text{S-TOP} \rightarrow \text{X}_1 \textit{ hiss} \mid \text{NP-SB}_1 \textit{ fauchen} \\ \langle \textit{fauchen } \text{CAT} \rangle = \texttt{VAFIN} \\ \langle \text{NP-SB INFL} \rangle = \langle \textit{fauchen } \text{INFL} \rangle \end{aligned}$$

7.4.4 Summary of Features

The lexical features and values used are summarised in Table 7.1. The target column indicates the parts of the phrase — determiner, attributive adjectives, noun, and verb — on which the information is marked. The source column states how the feature values are determined.

7.5 Training

We now describe how our model's lexicon and constraints can be derived from the training data. This is a straightforward rule-based procedure but it relies upon the availability of a statistical phrase-structure parser and morphological analyser to generate linguistic annotation of the data. These tools are readily available for German (we will touch on the availability of tools for other languages when we discuss the potential application of our approach to other languages in Chapter 10).

```
> wilde
wild<+ADJ><Pos><Fem><Nom><Sg>
wild<+ADJ><Pos><Fem><Akk><Sg>
wild<+ADJ><Pos><NoGend><Nom><Pl><St>
wild<+ADJ><Pos><NoGend><Akk><Pl><St>
wild<+ADJ><Pos><Masc><Nom><Sg><Sw>
wild<+ADJ><Pos><Neut><Nom><Sg><Sw>
wild<+ADJ><Pos><Neut><Akk><Sg><Sw>
```

Figure 7.1: Output of the Morphisto morphological analyser for the input word *wilde* (*fierce*).

7.5.1 Lexicon Extraction

Morphological analysis is a standard task in natural language processing and there are freely-available tools for extracting exactly the feature values that we are interested in. We use the Morphisto morphological analyser (Zielinski and Simon, 2009) in this work.

For each distinct target word in our training corpus, the morphological analyser produces a set of possible analyses. Figure 7.1 shows the analyses for the adjective *wilde*. Each analysis includes a lemma, a part of speech value, and a set of feature values. The features values are the same as those listed earlier in Table 7.1 except for superficial differences (like a single value `<Sw/Mix>` to indicate that the inflection is consistent with both the weak and mixed adjective declension paradigms).

The part of speech values are similar but not identical to those used in the Tiger corpus (and output by the BitPar statistical parser). Generally speaking, the categories used by the morphological analyser are coarser-grained but the same fine-grained distinctions are instead encoded as feature values. We use a simple mapping scheme to produce the Tiger-compatible CAT values for our lexicon. Full details are provided in Appendix A.

7.5.2 Constraint Extraction

In our baseline system, the rule extraction step (described in Section 6.2.3) uses the GHKM algorithm to extract an SCFG translation grammar. We extend rule extraction to generate the same grammar but with a set of constraints for each SCFG rule. This involves the following steps:

1. Tree annotation. The syntax of the German parse trees is used to match selectors

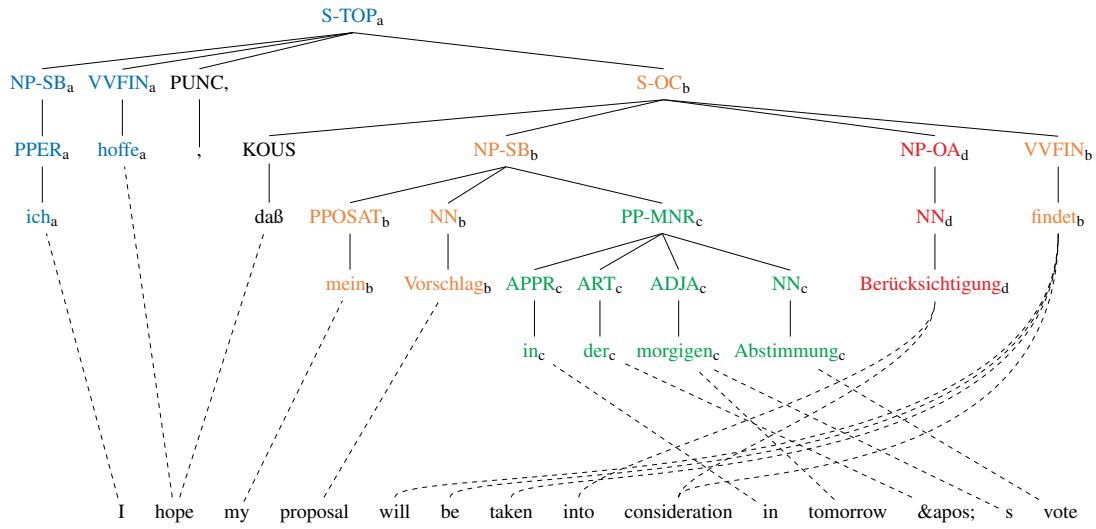


Figure 7.2: Alignment graph for sentence pair from training data. The target sentence has four selector-target sets, indicated by colour (and by the subscripts a, b, c, and d).

(nouns and prepositions) with their targets (determiners, attributive adjectives, nouns, and finite verbs) and label the participating nodes in each selector-target set.

2. Identity generation. Rule extraction is extended to generate identities between feature values whenever an SCFG rule contains two or more nodes from a common selector-target set. CAT identities are added for terminals that appear in other identities to allow for disambiguation of lexicon entries based on part of speech.

We now outline how these two steps are performed.

Annotation of Selector-Target Sets

Figure 7.2 shows a sentence-pair from the training data with the selector-target sets indicated using colour. In our annotation scheme, a selector-target set contains one or more selectors and target words, their lowest common constituent node, and any intermediate constituent nodes.

If a target word is selected by two distinct selectors, a single selector-target set containing both selectors is formed. For example, the determiner *der* in the prepositional phrase is case-governed by the preposition *in* and controlled by the noun *Abstimmung*. All three words are added to the same set (along with the selectors' other target, the adjective *morgigen*).

The Tiger treebank annotation allows for identification of selector-target relationships based on simple constituency patterns. Whilst there are a few tricky cases to deal with, most relationships can be identified from a few syntactic patterns and so we use a simple rule-based procedure. The procedure is purely syntactic (lexical content is ignored) and includes rules like "a word is a noun phrase head (and therefore a controller) if its preterminal label is one of NN, NE, PPER, or PDS, its grandparent's category is NP, there are no commas among its left siblings, and there are no nouns or pronouns among its right siblings (unless they are preceded a comma)." The full set of rules is given in Appendix B.

Induction of Constraints

Recall from Section 3.2.2 that in GHKM each grammar rule r is derived from a subgraph h of the alignment graph. If r is written

$$\mathrm{Y}_0 \rightarrow \mathrm{X}_1\ \mathrm{X}_2 \ldots \mathrm{X}_m \ \mid\ \mathrm{Y}_1\ \mathrm{Y}_2 \ldots \mathrm{Y}_n$$

then the head symbol Y_0 is projected from the root node of h and the target body symbols, $\mathrm{Y}_1 \ldots \mathrm{Y}_n$, are projected from its sink nodes. For a rule symbol Y_i we will write h_i to denote the projecting node of the subgraph. If Y_i is a terminal we will write $\mathrm{pos}(i)$ to denote the part-of-speech label from h_i's parent node. If h_i is a NP or is dominated by a NP then we will write $\mathrm{np}(i)$ to denote the lowest dominating NP.

The constraints for each rule r are generated as follows:

1. For each pair i and j, $i < j \leq n$, for which h_i and h_j belong to a common selector-target set S and where i is the least value such that $h_i \in S$, the following identity is a constraint: $\mathrm{Y}_i(\textsc{infl}) = \mathrm{Y}_j(\textsc{infl})$.

2. If $\mathrm{Y}_i(\pi)$ is a constraint term and Y_i is a terminal then $\mathrm{Y}_i(\textsc{cat}) = \mathrm{pos}(i)$ is also a constraint.

3. If $\mathrm{Y}_i(\pi)$ is a constraint term, h_i belongs to a selector-target set S that does not contain a determiner, and i is the least value such that $h_i \in S$ then $\mathrm{Y}_i(\textsc{decl}) =$ `strong` is also a constraint.

4. If $\mathrm{Y}_i(\pi)$ is a constraint term and h_i is either an NP or is dominated by a NP, and if the syntactic category of $\mathrm{np}(i)$ has a predominant case value c then $\mathrm{Y}_i(\textsc{infl case}) = c$ is also a constraint.

In the resulting constraint set, every constraint is associated with exactly one selector-target set since i) every constraint includes at least one non-constant constraint term $\text{Y}_i(\pi)$, ii) the corresponding node h_i must belong to a selector-target set, and iii) if the constraint includes a second non-constraint constraint term $\text{Y}_j(\pi)$ then h_j must belong to the same selector-target set as h_i. This property means that the constraint set can be partitioned such that there is one subset of constraints for each selector-target set.

Whilst constraint induction has been described using the terminology of GHKM, the same process can be used for Hiero-based string-to-tree rule extraction methods (as was the case in Williams and Koehn (2011)).

Example

As an example, Figure 7.3 shows the alignment graph for a sentence pair from the training data. There are two selector-target sets (indicated in blue and orange, and with the subscripts a and b). The root and sink nodes of one possible subgraph are shown using boxes. This subgraph is one of many that can be formed by composing minimal GHKM rules.

Figure 7.4 shows the SCFG grammar rule that is extractable from this subgraph along with the constraints that are induced. The constraint set can be partitioned into two subsets (indicated by the dashed line) where the constraints of each correspond to one of the two selector-target sets.

The case constraints, a_5 and b_5, result from the fact that NPs with the syntactic categories SB and AG are predominantly in nominative and genitive case, respectively.

Partial Coverage of Selector-Target Sets

In the example just given, the rule is produced from a large subgraph that fully incorporates the two selector-target sets. Note that the inclusion of a complete selector-target set is not a precondition for the extraction of effective constraints since unification facilitates the propagation of relevant information from smaller derivations: for example, a subgraph covering only the NP-SB constituent would result in a rule with constraints that propagate agreement information to the root. A second rule derived from a subgraph containing the S-TOP but with NP-SP as a sink node would have a constraint that unifies the information from NP-SP subderivation with that of the verb.

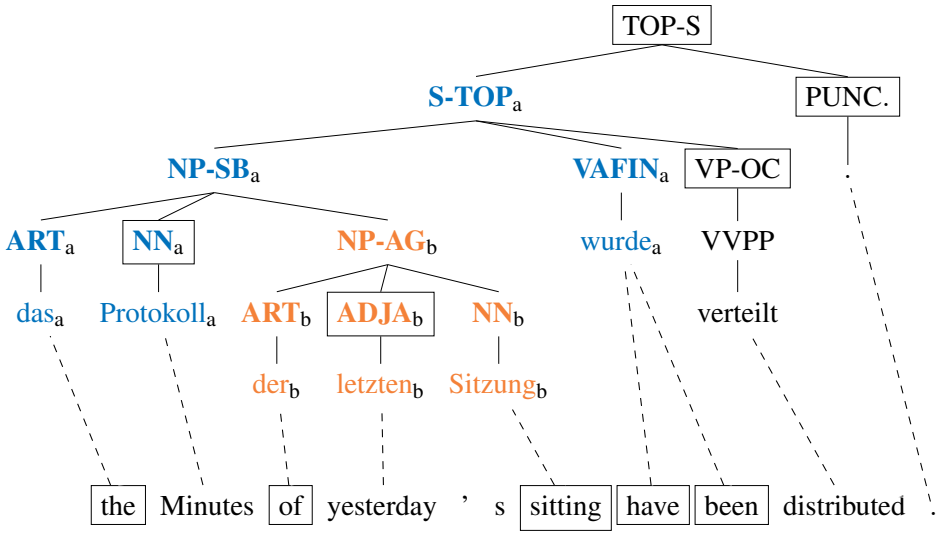


Figure 7.3: Alignment graph for sentence pair from training data. The boxes indicate the root (TOP) and sink nodes of a subgraph from which a composed rule is extractable (as defined by the GHKM algorithm).

$$\begin{aligned}
&\text{TOP} \rightarrow \textit{the}\ \text{X}_1\ \textit{of}\ \text{X}_2\ \textit{sitting have been}\ \text{X}_3\ \text{X}_4\ \mid \\
&\qquad \textit{das}\ \text{NN}_1\ \textit{der}\ \text{ADJA}_2\ \textit{Sitzung wurde}\ \text{VP-OC}_3\ \text{PUNC}._4 \\
&\langle \textit{das}\ \text{INFL} \rangle = \langle \text{NN}\ \text{INFL} \rangle && (a_1) \\
&\langle \textit{das}\ \text{INFL} \rangle = \langle \textit{wurde}\ \text{INFL} \rangle && (a_2) \\
&\langle \textit{das}\ \text{CAT} \rangle = \texttt{ART} && (a_3) \\
&\langle \textit{wurde}\ \text{CAT} \rangle = \texttt{VAFIN} && (a_4) \\
&\langle \textit{das}\ \text{INFL CASE} \rangle = \texttt{nom} && (a_5) \\
&\text{- - - -} \\
&\langle \textit{der}\ \text{INFL} \rangle = \langle \text{ADJA}\ \text{INFL} \rangle && (b_1) \\
&\langle \textit{der}\ \text{INFL} \rangle = \langle \textit{Sitzung}\ \text{INFL} \rangle && (b_2) \\
&\langle \textit{der}\ \text{CAT} \rangle = \texttt{ART} && (b_3) \\
&\langle \textit{Sitzung}\ \text{CAT} \rangle = \texttt{NN} && (b_4) \\
&\langle \textit{der}\ \text{INFL CASE} \rangle = \texttt{gen} && (b_5)
\end{aligned}$$

Figure 7.4: SCFG grammar rule and induced constraints for the subgraph in Figure 7.3.

Tag	Meaning	Nom	Acc	Dat	Gen	Total	Relative
SB	Subject	99.8	0.0	0.1	0.1	1,316,357	46.7%
AG	Genitive attribute	0.1	0.0	0.1	99.8	775,305	27.5%
OA	Accusative object	1.2	95.5	2.9	0.4	258,307	9.2%
DA	Dative	0.9	0.6	97.5	1.0	161,518	5.7%
HD	Head	7.9	11.1	67.8	13.2	99,389	3.5%

Table 7.2: NP case frequencies for the five most common syntactic relation tags.

Case Frequencies

Case constraints are added for rule elements that belong to NPs with syntactic categories for which there is one predominant grammatical case. For many of these categories, the appropriate case is clear from the definition of the category used in the treebank annotation. For example, the AG category is used to indicate a 'genitive attribute.' For other categories there may not be a clear choice.

Rather than relying on treebank definitions, we determine empirically when a syntactic category label can be relied upon to indicate a single case. To do this, we annotate the target-side of our training corpus to indicate selector-target set membership, as we have just described. We also extract a feature structure lexicon, again using the method just described. We then induce and evaluate constraints for the smallest possible composed rule that fully covers each selector-target set. For each NP, we record a count for the case value occurring with the NP's syntactic category label. If there are multiple possible case values then we divide the counts between the values. Table 7.2 shows the case frequencies for the five most frequently occurring NP syntactic category labels. During training, we use the frequencies to determine whether or not to induce a case constraint. We use a threshold of 95%.

Dealing with Conflicting Constraints

Our constraint extraction method relies on syntactic annotation from a statistical parser. However good we make the procedure for annotating selector-target sets, there will always be a danger that erroneous constraints are learned due to parse errors. For example, if the parser were to misidentify the definite article of a noun phrase as a relative pronoun — in German, some forms, like *der* and *die*, function in both roles — then grammar rules derived from the (presumed-determiner-less) noun phrase would

gain a spurious strong declension constraint.

As a guard against this kind of error, we test for conflicting constraints during training: for each set of rules that share a common target-side (that is, a pair (C, α) where C is the rule's left-hand-side non-terminal and α is the target right-hand-side), we count the distinct constraint sets and retain the most frequent set, provided it has a relative frequency of 0.7 or higher.[2] The chosen constraint set is then used for all rules with that target-side. If the most frequent constraint set does not meet the relative frequency threshold then constraints are dropped from those rules.

7.6 Experiments and Analysis

To evaluate the effect of our agreement and government constraints we compare the baseline system from Chapter 6 with three systems. The first is identical to the baseline except that BitPar's morphological tags were retained during training and included in the translation grammar's non-terminal labels. During the early development of the baseline, we had found that including the morphological tags degraded translation quality and so they were stripped from the parse tree labels prior to rule extraction. As we see shortly, we now observe a small improvement in BLEU from retaining them and so we include this system for comparison. The other two systems use constraints, one with hard constraints and one with soft constraints. For the soft constraint model, we add a single feature function: a count of the number of constraint evaluation failures.

7.6.1 BLEU

Following the recommendation of Clark et al. (2011), we ran the MERT optimization step three times for each system and repeated evaluation with each set of feature weights. Table 7.3 shows the averaged single-reference BLEU scores and standard deviations.

Contrary to our earlier results (Williams and Koehn, 2011), we find that hard constraints perform better than soft constraints. In that work, we suggested that our constraint extraction heuristics may be introducing significant numbers of spurious constraints and that using soft constraints allowed the decoder to overcome these defi-

[2] This value was chosen during system development by decoding the dev set with hard constraints obtained using various settings. In practice, the value was found to have a small effect on resulting BLEU scores and a wide range of settings produced near-identical results. Once the value had been chosen it was not re-tuned.

System	2009		2010		2011	
Baseline	15.1	0.1	16.6	0.1	15.4	0.2
Morph tags	15.3	0.0	16.6	0.1	15.5	0.1
Soft constraint	15.4	0.0	16.9	0.0	15.6	0.0
Hard constraint	15.6	0.0	17.1	0.1	15.7	0.1

Table 7.3: Average BLEU scores and standard deviations over three optimization runs.

		s-BLEU			
		Worse	Same	Better	Total
Model score	Worse	12.8	13.1	17.8	43.7
	Same	0.0	53.0	0.0	53.0
	Better	1.1	1.0	1.3	3.4
	Total	13.9	67.0	19.1	

Table 7.4: Effect on model score and s-BLEU of using hard constraints (values are percentages of sentences, calculated over the three test sets).

ciencies by permitting some constraint failures. Since that work we have substantially improved the tree annotation scheme and numerous other aspects of training and so we believe that softening the constraints is less helpful (we provide a summary of the main training improvements at the end of this section).

Sentence-Level Analysis

To get a clearer picture of the effect our constraints were having, we ran a system with and without hard constraints using identical tuning weights. We used the weights from the first baseline tuning run. This comparison removes optimization-related noise from consideration, making analysis of individual sentence-level changes meaningful, though it disadvantages the hard constraint system by using non-optimized weights.

Table 7.4 shows the effect of using hard constraints on model score and sentence-level BLEU (s-BLEU). The results from the three test sets were aggregated.

One possible advantage of using hard constraints is that removing ungrammatical subderivations early in the search allows for greater diversity, ultimately improving search accuracy. Based on the low proportion (3.4%) of sentences for which the model score improves, this appears to be a minor effect. A far larger proportion of sentences

System	2009-20	2010-20	2011-20
Baseline	16.8	17.1	14.5
+noun-modifier	17.0	17.5	14.7
+prep-gov	17.0	17.5	14.7
+adj-decl	17.0	17.6	14.7
+np-case	17.1	17.8	14.9
+subj-verb	17.2	17.8	14.9

Table 7.5: BLEU scores for short sentences as constraint types are progressively included.

(43.7%) have a worse model score, and just over half are unchanged.

As we would expect from the increase in test-set BLEU, the number of sentences for which s-BLEU increases (19.1%) outweighs the number for which it decreases (13.9%). In other words, according to s-BLEU, for every sentence that translation quality is degraded, there are approximately 1.4 for which it is improved.

Of the sentences that change when hard constraints are used, s-BLEU gives identical scores for approximately 29.8%. Since the agreement and government constraints leave many words unchanged, it is unsurprising that BLEU will fail to detect some changes: replacing a word that does not occur in the reference with another word that does not occur in the reference will leave the score unchanged.

Contribution of Individual Constraint Types

To measure the contribution of individual constraint types, we progressively added constraint types to the baseline, measuring the BLEU score at each point. To eliminate optimization noise, we used a single set of baseline weights and did not re-tune. We used constrained versions of the test set in this experiment, including sentences up to 20 tokens only. During development, we found our constraints to be more effective on shorter sentences and since the BLEU changes are already small, this shows up the differences more clearly.

Table 7.5 shows the results. The constraint types follow the description in Section 7.4: noun-modifier agreement, prepositional case government, adjectival declension types, NP case marking, and subject-verb agreement.

7.6.2 Human Evaluation

Whilst BLEU remains the most widely used automatic metric for statistical machine translation, its limitations are well documented (see Callison-Burch et al. (2006) for a critique of BLEU and see the many proposals for alternative metrics). Given BLEU's limitations, human evaluation remains an essential part of translation competitions and is the basis of the offical WMT system rankings (Bojar et al., 2013).

To perform a similar evaluation, we asked two native German speakers to compare the translation quality of the baseline and hard constraint systems (again, re-using baseline weights). From the 45% of sentences that changed between baseline and hard-constraint systems we removed sentences longer than 20 tokens (leaving 1,203) and then generated a random sample of 300 sentences (without duplicates). We asked two annotators to judge which translation they preferred. We restricted our sample to short sentences because they tend to be easier to compare.

Figure 7.5 shows a screenshot of the web interface that was used by our annotators. The order of the two output sentences was randomly chosen (and the same for both annotators). If an annotator thought the two translations were equally good (or equally bad) then they had the option to state that they did not have a preference. We did not suggest any judgement criteria and the annotators were not told anything about the translation systems. The results are shown in Table 7.6.

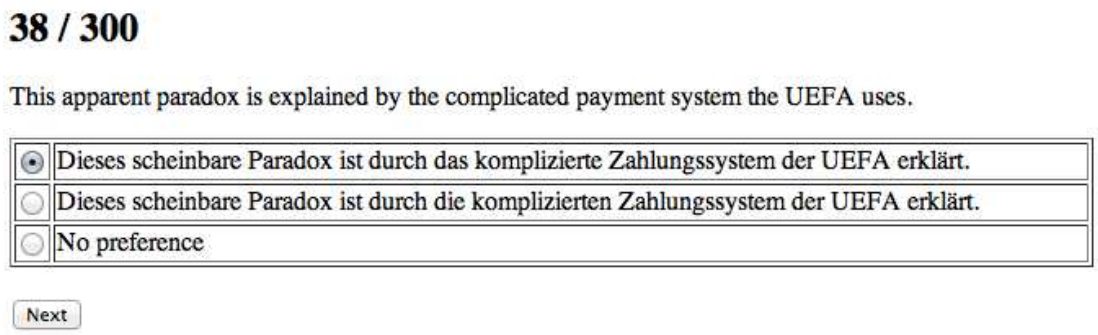


Figure 7.5: Screenshot of the web interface used for the human evaluation task.

Following the advice of Carletta (1996) and the example of the WMT evaluation task (Bojar et al., 2013), we used Cohen's kappa coefficient (Cohen, 1960) to calculate a measure of inter-annotator agreement. The kappa coefficient is defined as

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)}$$

		Annotator A			
	Judgement	Worse	Same	Better	Total
Annotator B	Worse	31	15	8	18.0%
	Same	28	33	77	46.0%
	Better	8	12	88	36.0%
	Total	22.3%	20.0%	57.7%	

Table 7.6: Judgements for the 300 sentences by the two annotators.

where $P(A)$ is the proportion of sentences on which the annotators agree and $P(E)$ is the proportion of sentences that we would expect the annotators to agree on by chance.

The agreement value is low ($\kappa = 0.253$). It is clear from Table 7.6 that the annotators have different views on when a translation difference is qualitatively worse or better or is the same. If we only include sentences for which the annotators both stated a preference one way or the other then this leaves 135 sentences for which the coefficient is much higher ($\kappa = 0.712$).

Recall that for the full test sets, s-BLEU indicated that for every sentence that translation quality is degraded, there were approximately 1.4 for which it improved. For the 300 sentence sample, s-BLEU gives a higher ratio of 1.6 (the worse/same/better percentages were 27.0%, 29.3%, and 43.7%). The ratios for the two annotators are higher at, respectively, 2.0 and 2.6 improved sentences for every degraded one.

7.6.3 Translation Examples

We now give some examples of the changes that our constraints produce. In the first (Figure 7.6), there is a change of case for the object (*ein Kerl* versus *einen Kerl*) and a change in noun from one synonym to another (*Jacke* versus *Jackett*) to match the inflection. Both annotators prefer the hard constraint output, and s-BLEU scores it higher:

Input	I saw a guy of about thirty with a green jacket.
Ref.	Ich habe einen Kerl in den Dreissigern mit einer grünen Weste gesehen.
Baseline	Ich sah ein Kerl von etwa 30 mit einem grünen Jacke.
HC	Ich sah einen Kerl von etwa 30 mit einem grünen Jackett.

Figure 7.6: Sentence 1232 from the newstest2010 test set.

In the next example (Figure 7.7), the inflection of the verb changes to match the

subject. s-BLEU scores the baseline higher, whereas both annotators prefer the hard constraint output:

Input	The minimum width for one should be 90cm in double bed, i.e. 180 cm altogether.
Ref.	Jedem der Schläfer sollten mindestens 90 cm Liegebreite zur Verfügung stehen, zusammen also 180 cm.
Baseline	Die Mindestbreite für einen sollten im Ehebett 90cm sein, d. h. 180 cm.
HC	Die Mindestbreite für einen sollte im Ehebett 90cm sein, d. h. 180 cm.

Figure 7.7: Sentence 210 from the newstest2011 test set.

The reverse is true in the third example (Figure 7.8): the human annators prefer the baseline, but the hard constraint output receives a better s-BLEU score. The inflection of the hard constraint system is correct, unlike the baseline, but it involves a poorer choice of noun:

Input	Cockell was not part of the research team.
Ref.	Cockel war nicht Teil des Forschungsteams.
Baseline	Cockell war nicht Teil der Forscherteam.
HC	Cockell war nicht Teil des Teams der Forschung.

Figure 7.8: Sentence 1395 from the newstest2011 test set.

In the next example (Figure 7.9), the choice of determiner is changed to agree with the noun. The human annotators prefer the change, but the s-BLEU score does not change:

Input	Without an excessive ego, the ultimate mark of quality.
Ref.	Ohne übermäßiges Ego, äußerstes Qualitätsmerkmal.
Baseline	Ohne eine übermäßige Eitelkeit, die ultimative Zeichen der Qualität.
HC	Ohne eine übermäßige Eitelkeit, das ultimative Zeichen der Qualität.

Figure 7.9: Sentence 1824 from the newstest2009 test set.

Whereas the previous examples have involved changes over a short distance, the final example (Figure 7.10) illustrates a change that is well beyond the range of a typical *m*-gram language model. In this example, the verb inflection (from *könnte* to *könnten*) is changed to agree with the plural subject (the first *wir*), the choice of determiner is changed to agree with the noun. One human annotator prefers the change whilst the other does not have a preference. The s-BLEU score does not change.

Input	The only threat was that if we don't stop this, we could lose our bus licence.
Ref.	Die einzige Drohung war, wenn wir darauf nicht verzichten, dass wir dann auch die Buslizenzen verlieren können.
Baseline	Die Bedrohung war nur, dass wir, wenn wir das nicht stoppen, unseren Bus Lizenz verlieren könnte.
HC	Die Bedrohung war nur, dass wir, wenn wir das nicht stoppen, unseren Bus Lizenz verlieren könnten.

Figure 7.10: Sentence 447 from the newstest2010 test set.

7.6.4 Computational Costs

Constraint evaluation adds computational cost to decoding, which we have measured empirically for our model. We compared the baseline system to the systems with hard and soft constraints by decoding the same test set with each system and then comparing translation time and peak memory usage.

The test set was sampled from the newstest2009-2011 test sets by randomly choosing (without replacement) ten sentences of length 1-10, ten of length 11-20, and so on. We decoded the test set four times for each system, discarding the first set of results (to allow for filesystem cache priming) and then averaging the remaining three. The decoder was run in single-threaded mode in order that we could obtain accurate decoding times for individual sentences.

Table 7.7 shows the total decoding times for each system and the peak virtual memory usage. The test machine had 48GB of physical memory, which was more than sufficient for the processes to run without swapping out to disk. The machine was lightly loaded, which is reflected in the low variance in decoding times between system runs. Since the decoding algorithm is deterministic, peak memory usage is effectively constant for runs of the same system and so variance is not reported (the largest difference was less than 0.5MB).

Figure 7.11 shows plots of sentence length against decoding time for the three systems. The empirical exponents were calculated by using the least-squares method to fit a straight line to the data-points (in log space). The resulting curves are shown along with their exponents.

System	Time (s)	Δ	s.d.(s)	VM (MB)	Δ
Baseline	7,507	-	20.5	7,744	-
Hard constraint	8,765	+16.7%	12.4	10,649	+37.5%
Soft constraint	10,315	+37.4%	14.6	10,966	+41.6%

Table 7.7: Total decoding time and peak virtual memory usage, averaged over three runs. The percentage changes relative to the baseline are shown, as are the standard deviations.

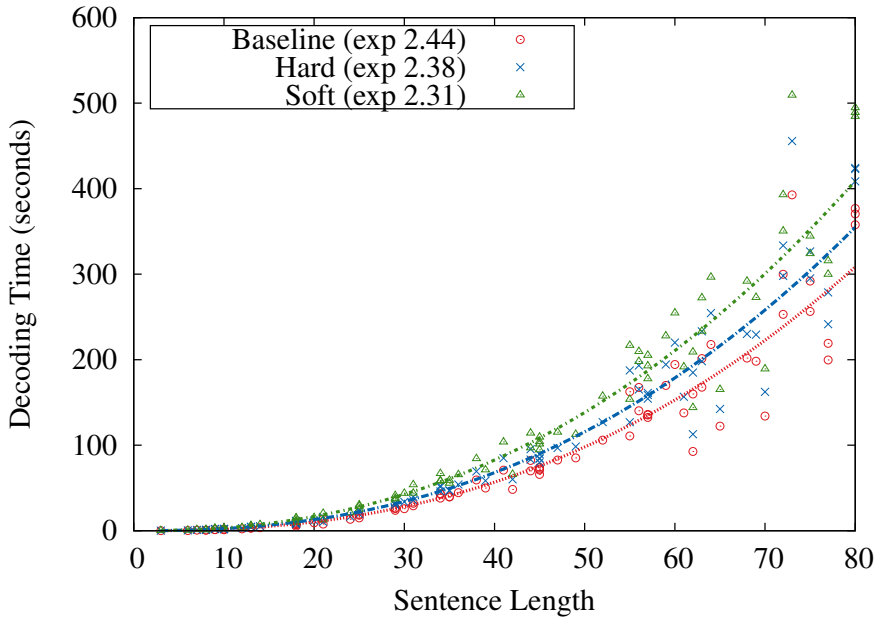


Figure 7.11: Sentence length vs decoding time for the baseline, hard constraint, and soft constraint systems.

7.6.5 Comparison to Williams and Koehn (2011)

In an earlier version of this work (Williams and Koehn, 2011) we found soft constraints to be more effective than hard constraints, whereas we now find the opposite. A direct comparison of results is not possible due to changes in experimental setup (the experiments in Williams and Koehn (2011) used less training data (from 2011's WMT shared task rather than 2012's) and were only evaluated on short sentences). However, the training process did evolve considerably during the development of this thesis and we think it is plausible that a cumulation of training improvements is the main reason for the apparent superiority in hard constraints (although we believe that both the soft and hard constraint models could be improved further, as we will discuss in Chapter 10). We summarise the main changes to the training process below.

- We switched from using a Hiero-based rule extraction method to using GHKM with scope-3 pruning. This had two benefits: it led to an improvement in baseline translation quality and it removed the reliance on glue rules (as reported in Williams and Koehn (2012)). We observed in Williams (2009) and during early development of this model that the decoder would frequently circumvent constraints by using glue rules or producing syntactically ill-formed constructions.

- The tree annotation procedure (Section 7.5.2 and Appendix B) is more accurate than that of Williams and Koehn (2011). This was measured indirectly during development by inducing and evaluating constraints (in the same way that we did for case frequencies) for a portion of the training data. A lower failure count was taken to indicate an improvement.

- Unlike in Williams and Koehn (2011), case constraints are not probabilistic, but are absolute (and only used where a single choice of case value has an observed relative frequency that surpasses a high threshold). Whilst this may seem like a disadvantage, it is a more natural fit for the most frequent syntactic relation tags (such as subject and genitive attribute) where only one case value is acceptable. During development, the latter approach was found to work better than the former. It also has the benefit of removing a feature weight that must otherwise be learned during tuning.

7.7 Conclusion

In this chapter we have shown how our framework can be applied to the task of modelling agreement and government phenomena, and how such a model can be trained. We used German as our target language, but similar phenomena occur in most Indo-European languages.

Compared with previous work on generating target inflection in SMT, our approach allows for inflection patterns to be modelled in terms of constituent structure and without restriction on the range of relationships. Our model directly links syntax (in the grammar rules) and morphological features (in the lexicon) via constraints.

We showed empirically that our approach improves translation quality. Narrowly-focused changes are difficult to measure using BLEU and so we also used human evaluation. We found that the human annotators had a stronger preference for our system over the baseline than was reported by s-BLEU (a reported ratio of 2.0 or 2.6 improved sentences to each degraded one, compared with 1.6 measured by s-BLEU).

Whereas many of the observed improvements occur over short distances and could conceivably be obtained through the use of a better *m*-gram language model, the combinatorial nature of language means that a *m*-gram language is always likely to encounter combinations of words that were not seen during training, and this is particularly true for languages with rich inflectional morphology. For Arabic, the bi-gram agreement model of Green and DeNero (2012) provides further evidence for the effectiveness of models that are complementary to the *m*-gram language models. For Czech, Ondřej Bojar (personal correspondence) reports that agreement errors at short distance are common even using a *m*-gram model trained on 3.6 billion words.

Chapter 8

Verbal Complex Production

. . . finally, all the parentheses and reparentheses are massed together between a couple of king-parentheses, one of which is placed in the first line of the majestic sentence and the other in the middle of the last line of it – after which comes the VERB, and you find out for the first time what the man has been talking about; and after the verb – merely by way of ornament, as far as I can make out – the writer shovels in "haben sind gewesen gehabt haben geworden sein," or words to that effect, and the monument is finished. — Mark Twain (1880)

8.1 Introduction

A verbal complex is a main verb and its accompanying auxiliary verbs taken as a (possibly discontinuous) unit. This chapter investigates the problem of producing verbal complexes in translation.

Like agreement and government, accurate production of verbal complexes involves aspects of both syntax and inflectional morphology, with coordination over multiple target words. There are some parallels with the phenomena studied in the previous chapter that suggest feature structures and unification are well-suited to the task of modelling verbal complex production. In the previous chapter, each word in a selector-target set contributed partial information to a shared INFL structure — an abstract representation of inflectional features. A lexicon was used to list the set of valid INFL structures for each target form and, via the constraints, to block ungrammatical combinations of forms. Similarly, each individual verb form in a verbal complex can be thought of as contributing partial information to a single abstract verbal complex structure. Some combinations of forms are consistent whilst others are not, and a feature structure lexicon provides a means of specifying the valid types of verbal complex

structure that are possible through combination (thus blocking non-grammatical combinations).

We begin this chapter with a discussion of why verbal complex production is a challenging problem for statistical machine translation, particularly for languages like German and Dutch that involve long-range separation of auxiliaries and main verbs. We then develop a representation of German verbal complexes as feature structures, using constraints to ensure that consistent values are produced during translation. By testing for failures in verbal complex production, we identify where errors occur in our baseline system and present a detailed manual analysis.

We find that using constraints alone leads to little improvement over the baseline, but our analysis confirms that the method is effective at identifying translation errors. This allows us to quantify the errors and shows that they occur frequently enough for the problem to be worth addressing. The analysis also suggests how to proceed and in the next chapter we will extend the model to use source-side information to influence the selection of verbal complex types during search.

8.2 Background

8.2.1 Verbal Complexes

Many languages use multi-verb constructions in which a main content verb, possibly with associated particles, is combined with one or more auxiliary verbs. For example, in English we can say ‘is producing,’ ‘will have been produced,’ and ’should produce.’ The inflection of the main verb and the choice of auxiliary verbs jointly express grammatical properties that are crucial to meaning, such as tense, mood, and voice. The inclusion of particles usually changes the semantics of the verb (for example, ‘pick up’ instead of ‘pick’). In this thesis, we adopt the terminology of Gojun and Fraser (2012) and use the term ‘verbal complex’ to mean a main verb and any associated auxiliaries and particles within a single clause.

In English, a verbal complex can be discontiguous. Typically, this is due to the insertion of adverbs separating the auxiliaries and the main verb (for example, in ‘is carefully playing’ and ‘is once again playing’). In Dutch and German, the auxiliary and main verb can appear at opposite ends of a clause, separated by an arbitrarily-long sequence of arguments and adjuncts.

8.2.2 Verbal Complexes and SMT

Correctly producing verbal complexes is a difficult task for SMT. For some constructions, a word-for-word translation of each source verb produces a reasonable translation — for instance, translating the English 'have played ...' to the German 'habe ... gespielt' by independently translating 'have' to 'habe' and 'played' to 'gespielt' — but in general there is not a one-to-one equivalence between individual source and target verbs.

Syntax-based translation models are able to learn discontiguous rules and so in principle they can learn rules that capture complete verbal complex translations, but in practice the resulting verbal complexes are often garbled, incomplete, or missing altogether. There are multiple routes by which ill-formed constructions come to be licensed by the model. We highlight two prominent sources of error: automatic word alignment and overly-strong independence assumptions implicit in the treebanks from which the syntactic annotation is derived. These underlying problems are certainly not specific to verbal complex production and are themselves the subject of active research.

Word Alignment Errors

Automatic word alignment, which is typically based on the IBM word-based models of Chapter 2, can introduce errors into rule extraction. For instance, Figure 8.1 shows an example from the training data in which a missing alignment link (between *has* and *ist*) allows the extraction of a rule that translates *has failed* to the incomplete *fehlgeschlagen*.

Figure 8.2 shows an example of a similarly incomplete rule being used by our baseline system. The input contains a single verb 'read' which is translated in the reference to 'habe ... gelesen.' In the baseline system, the auxiliary verb 'habe' is correctly produced, but the main verb is missing. 'Habe' can be used as a full verb in its own right and can occur in combination with verbs other than past participles, making interpretation of the incomplete output harder still for readers. The system is able to produce this derivation because the grammar contains the rule VAFIN $\rightarrow$ *read* | *habe*. In the training example from which the rule is learned, the translation is 'habe ... gelesen' but the automatically-learned word-alignment is missing a link between 'read' and 'gelesen.'

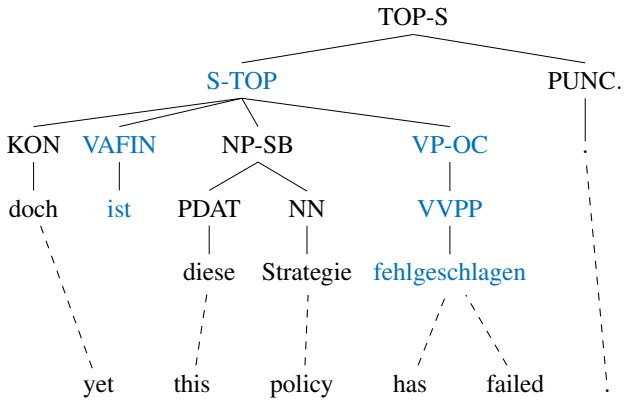
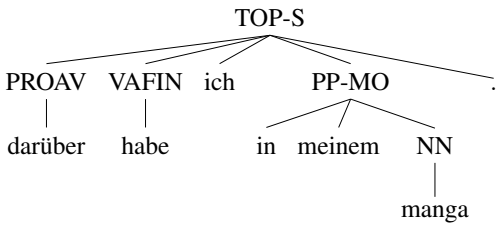


Figure 8.1: Alignment graph for a sentence pair from the training data. The target sentence has a single verbal complex comprising a main verb, *fehlgeschlagen*, and an auxiliary verb, *ist*.



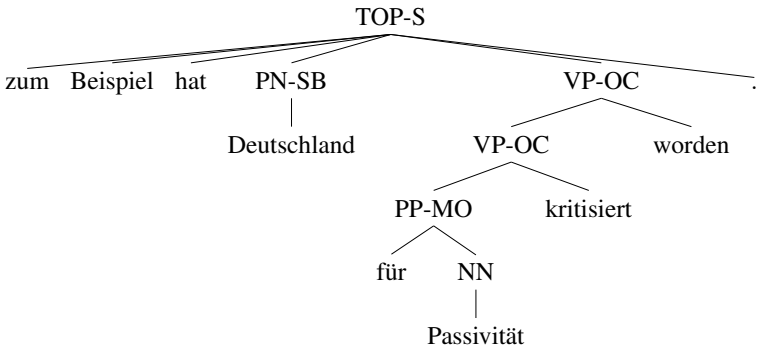
Input	I *read* about that in my manga .
Reference	davon *habe* ich in meinen Manga-Comics *gelesen* .
Gloss	about-it *have* I in my Manga-comics *read* .

Figure 8.2: 1-best translation of a sentence from the newstest2009 test set. The translation contains the auxiliary 'habe' but is missing the main verb.

Treebank-based Independence Assumptions

Even with perfect word alignments, the automatically extracted rules of a synchronous grammar may not include sufficient context to ensure the overall grammatically of a derivation. The extent of this problem will depend partly on the original treebank annotation style, which typically will not have been designed with translation in mind.

For instance, the Tiger treebank convention of attaching finite verbs to the clause node and embedding non-finite verbs in nested VPs may make the learning of rules containing complete verbal complexes more challenging than if, for example, all verbs were attached directly to the clause node. For grammar sizes to be practical, rule extraction algorithms place restrictions on rule size. Overly deep constituent structure



Input	for example , Germany *has been criticized* for passivity .
Reference	wegen Passivität *wurde* zum Beispiel Deutschland *kritisiert* .
Gloss	for passivity *was* for example Germany *criticized* .

Figure 8.3: 1-best translation of sentence 777 from the newstest2008 test set.

may fall foul of restrictions on node count and tree depth; on the other hand, overly flat constituent structure may pose problems where there are restrictions based on rule symbol counts.

Figure 8.3 shows a second example from our baseline system. In this example, the individual rules are reasonable in isolation, but the derivation contains an ungrammatical combination of verbs: for consistency with the choice of 'worden,' the finite verb should be a form of 'sein' not 'haben,' giving 'ist . . . kritisiert worden.' Even with this substitution, the less-literal translation in the reference, 'wurde . . . kritisiert,' would be more natural.

8.3 Previous Work

So far there has been little work on tackling verbal complex production as a problem in its own right. The wider problems of word alignment errors and treebank style have received more attention, though with different motivations.

8.3.1 Verbal Complexes

Gojun and Fraser (2012) tackle the problem of verbal complex translation in English-to-German in the context of phrase-based SMT. They address the fixed-window reordering limitation of phrase-based SMT by preprocessing the source-side of the training and test data to move English verbs within clauses into more 'German-like' posi-

tions.

Arora and Sinha (2012) consider a similar problem in English-Hindi translation. They improve a phrase-based model by merging verbs and associated particles into single tokens on both the source and target sides, thus simplifying the task of word alignment and phrase-pair extraction. Their approach relies upon the mostly-contiguous nature of English and Hindi verbal complexes. The discontiguity of verbal complexes rules out this approach for translation into German.

8.3.2 Word Alignment

There is a substantial body of work on improving word alignment for statistical machine translation. Specifically for syntax-based models, Wang et al. (2010) propose a syntax-based EM algorithm that realigns words after bootstrapping with GIZA++. However, they only find modest improvements in end-to-end translation quality. Fossum et al. (2008) train a supervised model that deletes word alignment links to promote the extraction of GHKM rules. They find that this leads to improvements in alignment link precision and BLEU score. Riesa et al. (2011) incorporates both source and target syntax into a discriminative word alignment model, improving both precision and recall compared to GIZA++.

8.3.3 Syntactic Annotation

Several approaches have been proposed for adapting treebank annotation for the specific task of statistical machine translation. Wang et al. (2010) proposes an EM algorithm to split or merge category labels on the target side of the training data prior to rule extraction and Chung et al. (2011b) applies a similar method to post-process the extracted synchronous grammar.

Automatic restructuring of trees has also been proposed, with the main motivation being to increase rule coverage. Wang et al. (2010) use an EM-based model to learn a tree binarization strategy and Burkett and Klein (2012) use a transformation-based error-driven learning approach that maximises the number of extractable rules.

8.3.4 Advantages of the Proposed Approach

It may be possible to improve verbal complex in our baseline system by re-implementing one or more of the approaches that we have just described. However, there are a num-

ber of reasons for adopting a constraint-based approach:

1. There is no inherent limit on the range of the verbal complex relationships that can be modelled. In phrase-based models, as have been used in previous work, the wide separation of main verbs and auxiliaries in German will often exceed the range of the *m*-gram language model and of any individual phrase-pair.

2. Our constraints can be targeted specifically to verbal complex translation. Whilst work on improving word alignment and tree annotation has been shown to improve overall translation and may indirectly improve verbal complex translation, its effect on any individual aspect of translation is hard to predict.

3. Our approach is largely orthogonal to the previous work. By taking a constraint-based approach, we are not precluding the later adoption of other approaches into our baseline model.

8.4 Model

We apply our approach to German, which has particularly challenging verbal complex forms due to the long-range separation of auxiliaries and main verbs. This section describes how we model verbal complex production using the lexicon and constraints of the formalism that was described in Chapter 5.

8.4.1 Feature Structures

A German verbal complex can be as small as a single word, as in *ich spiele* (“I play”) or *ich spielte* (“I played”), or it can involve up to four words, as in *es wird gespielt worden sein* (“it will have been played”). In all verbal complex forms, there is a single finite verb and zero or more non-finite verbs. The finite verb is inflected to indicate tense (past or present), mood (indicative or one of two subjunctive classes), and for agreement with the subject. The non-finite verbs are uninflected.

The grammatical properties of the finite verb are not necessarily the same as for the construction as a whole. For example, the future tense is expressed using an infinitive and a form of the auxiliary ‘werden’ in the present tense (similarly to in English, where ‘will’ and ‘shall’ are used to express the future tense).

Our feature structures are an abstracted representation of a verbal complex. They have two top-level features: FIN, which represents the finite verb, and NON-FIN, which

represents the non-finite part. The non-finite part may be empty or may involve one or more verbs.

As an example, Figures 8.4 and 8.5 show parse trees for two sentences from the training data together with the feature structure values that describe the verbal complexes. Apart from the internal NP structure, the syntactic structure of these two sentences is identical. In particular, they both have an auxiliary finite verb (labelled VAFIN) and a past-participle main verb (labelled VVPP). Within this syntactic configuration, the different lexical choices for the auxiliary verbs are used to express different types of grammatical voice (the first uses the *sein*-passive and the second uses the *werden*-passive).

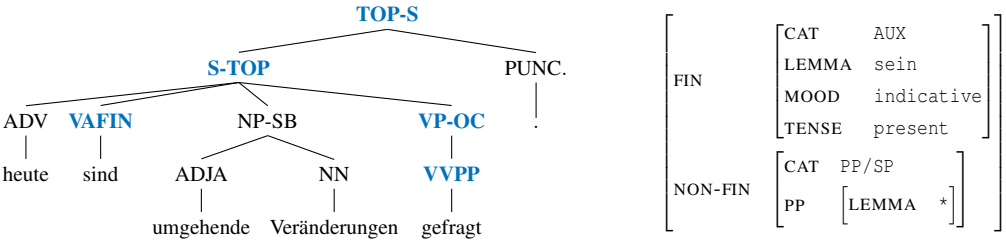


Figure 8.4: Sentence 3394 from Europarl. Gloss: today / is / urgent / change / required .

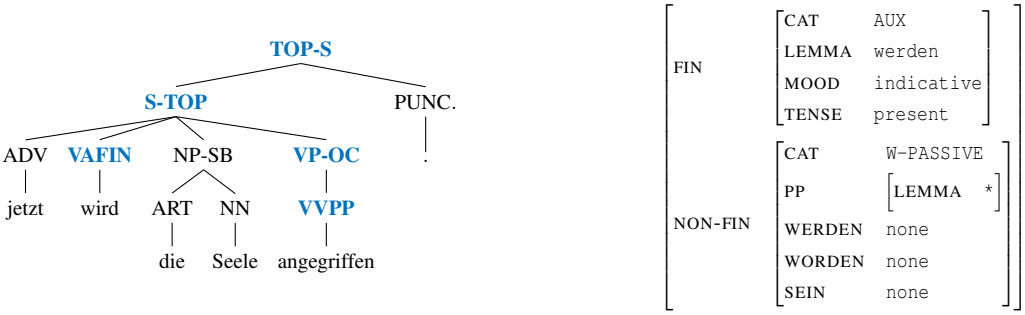


Figure 8.5: Sentence 4402 from Europarl. Gloss: now / is being / the / soul / attacked.

The FIN part of each feature structure value describes the properties of the auxiliary verb that contribute to the properties of the verbal complex as a whole. In this chapter, we do not model subject-verb agreement and so we abstract away number and person features.

The NON-FIN part describes the non-finite verbs that combine with the auxiliary. A *sein*-passive always includes a single past-participle. A *werden*-passive can include a

past-participle, as here, or a combination of fixed forms. The W-PASSIVE value indicates which of these are present. The lemma feature is used to distinguish between forms of *haben*, *sein*, and *werden*, and all other verbs (indicated with an asterisk).

Altogether, our feature structures use three different kinds of FIN value, of which AUX (used for auxiliary verbs) is one, and four kinds of NON-FIN value. Table 8.1 shows the mutually-exclusive categories to which the FIN and NON-FIN values can belong. The category of each FIN and NON-FIN value is indicated using a CAT feature.[1] The absence of a non-finite part is indicated by the use of the value none for the NON-FIN feature.

Feature	Category	Description
FIN	FULL	Full verb. Usually occurs without NON-FIN part
	AUX	Non-modal auxiliary verb. Requires NON-FIN part
	MODAL	Modal auxiliary verb. Requires NON-FIN part
NON-FIN	PP/SP	Non-finite part of perfect or sein-passive construction
	W-PASSIVE	Non-finite part of werden-passive construction
	INF	Infinitive, possibly with modal
	P-INF	Perfect Infinitive, possibly with modal

Table 8.1: Finite and non-finite sub-features

Table 8.2 shows the minimal pairs of finite and non-finite values that unambiguously express the six German tenses in the active voice. In effect, these minimal values constitute signatures for the active tenses. The optional addition of the mood feature to the finite verb refines the values along this grammatical dimension.

8.4.2 The Lexicon

We use the lexicon to specify the set of verbal complex feature structures that an individual verb can be a part of. For example, the entry for a past-participle includes feature structure values for *sein*-passive and *werden*-passive constructions, among others. Crucially, it does not include constructions in which there is no past-participle (such as future tense constructions).

[1] The CAT feature was suggested by the thesis examiners. In experiments, the lexicon used separate feature names for each category. The full set of three FIN features and four NON-FIN features was listed for each entry and a none value was used to indicate the values that weren't applicable. The two approaches lead to the same outcome, but the CAT approach is more elegant and readable, so we adopt it here.

Tense	**Finite Feature**	**Non-finite Feature**	**Example**
Present	[CAT FULL; TENSE present; LEMMA *]	none	ich laufe *I run*
Past	[CAT FULL; TENSE past; LEMMA *]	none	ich lief *I ran*
Perfect	[CAT AUX; TENSE present; LEMMA h/s]	[CAT PP/SP; PP [LEMMA *]]	ich bin gelaufen *I have run*
Pluperfect	[CAT AUX; TENSE past; LEMMA h/s]	[CAT PP/SP; PP [LEMMA *]]	ich war gelaufen *I had run*
Future	[CAT AUX; TENSE present; LEMMA werden]	[CAT INF; INF [LEMMA *]]	ich werde laufen *I shall run*
Future perfect	[CAT AUX; TENSE present; LEMMA werden]	[CAT P-INF; INF [LEMMA *]; HSINF [LEMMA h/s]]	ich werde gelaufen sein *I shall have run*

Table 8.2: Features corresponding to the six German tenses in the active voice. An example is shown for each along with its English equivalent. The lemma value `h/s` is an abbreviation for `haben/sein`.

For a multi-verb verbal complex, the lexicon includes a feature structure value for each participating verb. The values for the individual verbs are identical except for their FOUND values (described shortly) and for the mood feature, which depends solely on the inflection of the finite verb. The mood feature is specified for the relevant verb and left unspecified for the others.

For example, the entry for the infinitive *laufen* ('to run') contains the following feature structure, among others:

$$
\textit{laufen} \rightarrow \left[\begin{array}{ll}
\text{CAT} & \texttt{VVINF} \\
\text{VC} & \left[\begin{array}{ll}
\text{FIN} & \left[\begin{array}{ll}
\text{CAT} & \texttt{AUX} \\
\text{LEMMA} & \texttt{werden} \\
\text{FOUND} & []
\end{array}\right] \\
\text{NON-FIN} & \left[\begin{array}{ll}
\text{CAT} & \texttt{INF} \\
\text{INF} & \left[\begin{array}{ll}
\text{LEMMA} & * \\
\text{FOUND} & \texttt{true}
\end{array}\right]
\end{array}\right]
\end{array}\right]
\end{array}\right]
$$

The presence of this feature structure in the lexicon enables the form *laufen* to participate in active voice, future tense constructions such as *sie wird laufen* ("she will run") and *es würde laufen* ("it would run"). Note that the FIN value selects for forms of *werden* (which include *wird* and *würde*). The absence of similar entries for *laufen* with the LEMMA values `haben` and `sein` rule out ungrammatical verbal complexes like *habe . . . laufen.*

The participation of *laufen* in alternative tense/voice constructions, such as *wir laufen* ("we run") or constructions with modal finite verbs, such as *wir müssen laufen* ("we must run") is enabled by the presence of separate feature structures in *laufen*'s lexicon entry (the entry for *laufen*, like most other infinitives, contains a total of five feature structure values). Note also that variant verb forms, such as *laufe* or *gelaufen*, require entirely separate lexicon entries.

Whilst the multiplicity of feature structure values for individual forms can increase the size of the search space, the effect is mitigated by the choice of set-based search state representation adopted in Section 5.3.2: much of the ambiguity is contained within the search state and so does not affect pruning decisions. In later experiments (Section 8.6.2) we compare the effect on search in isolation and find the change in search quality to be minimal in practice.

We add a feature FOUND to indicate whether or not an individual verb is present. For each verb in the lexicon, the FOUND value in its feature structures is `true` for that verb and empty for any other verb that participates in the verbal complex. In combination

with a minor extension to the constraint mechanism, the FOUND values are used to ensure that the feature structures composed through unification are complete.

8.4.3 Constraints

We use constraints to ensure that VC values are compatible under unification. For example, the following rule requires that the clause's auxiliary and past-participle have compatible values:

$$\begin{aligned}&\text{S-TOP} \rightarrow \text{X}_1\ \text{X}_2\ \textit{required today} \mid \textit{heute}\ \text{VAFIN}_2\ \text{NP-SB}_1\ \textit{gefragt}\\ &\quad\langle \text{VAFIN VC}\rangle = \langle \textit{gefragt}\ \text{VC}\rangle\\ &\quad\langle \textit{gefragt}\ \text{CAT}\rangle = \texttt{VVPP}\end{aligned}$$

In order to ensure that the decoder does not produce incomplete verbal complexes (such as a past-participle without an auxiliary finite verb), we add a constraint that sets a flag indicating when a verbal complex is required to be complete:

$$\begin{aligned}&\text{S-TOP} \rightarrow \text{X}_1\ \text{X}_2\ \textit{required today} \mid \textit{heute}\ \text{VAFIN}_2\ \text{NP-SB}_1\ \textit{gefragt}\\ &\quad\langle \text{VAFIN VC REQUIRE-COMPLETE}\rangle = \texttt{true}\end{aligned}$$

The REQUIRE-COMPLETE constraints are added at the clause level where all parts of the verbal complex are present. We modify the decoder to perform the check for completeness. We do this by extending the EVAL-CONSTRAINTS algorithm (Section 5.3.3) to test the resulting VC values after constraint evaluation. If any FOUND value is not set to `true` within a VC value for which REQUIRE-COMPLETE is `true` then the VC value is discarded (as if it had failed constraint evaluation). This extension is enabled or disabled by a decoder configuration flag in order to allow for comparison.

The completeness condition here bears some similarity to the identically-named condition in LFG (Bresnan, 2001, p63). Both are well-formedness conditions that are applied to feature structures after the satisfaction of constraints. In LFG, completeness requires that every argument designated by a predicate feature be present in the feature structure containing the predicate.

8.5 Training

We now describe how our model's lexicon and constraints can be derived from the training data.

8.5.1 The Lexicon

Every verb observed in the training data has a non-empty entry in the lexicon. The lexicon's CAT values are derived from the parse trees on the target-side of the training data. The VC values are assigned according to CAT type from a small set of hand-written feature structures. For example, we saw the following feature structure for *laufen* earlier:

$$
\textit{laufen} \rightarrow \left[\begin{array}{ll}
\text{CAT} & \texttt{VVINF} \\
\text{VC} & \left[\begin{array}{ll}
\text{FIN} & \left[\begin{array}{ll} \text{CAT} & \texttt{AUX} \\ \text{LEMMA} & \texttt{werden} \\ \text{FOUND} & [] \end{array}\right] \\
\text{NON-FIN} & \left[\begin{array}{ll} \text{CAT} & \texttt{INF} \\ \text{INF} & \left[\begin{array}{ll} \text{LEMMA} & * \\ \text{FOUND} & \texttt{true} \end{array}\right] \end{array}\right]
\end{array}\right]
\end{array}\right]
$$

All verbs with the CAT value of VVINF are assigned this feature structure value as part of their lexicon entry. Every main verb is assigned one of three possible groups of VC values depending on whether the verb is finite, a past-participle, or an infinitive. The entries for the auxiliary and modal verbs are hand-written.

8.5.2 Constraint Extraction

The constraints are learned using a similar procedure to that used for agreement and government in Chapter 7:

1. Tree annotation. The syntax of the German parse trees is used to group verb nodes into sets, one for each verbal complex.

2. Identity extraction. Rule extraction is extended to generate identities between VC feature values whenever an SCFG rule contains two or more nodes from a common verbal complex set. CAT identities are added for terminals that appear in VC identities to allow for disambiguation of lexicon entries based on part of speech.

Annotation of Verbal Complex Sets

Figure 8.6 shows a sentence-pair from the training data with the verbal complex set highlighted. In our annotation scheme, a verbal complex set contains all the verbs of

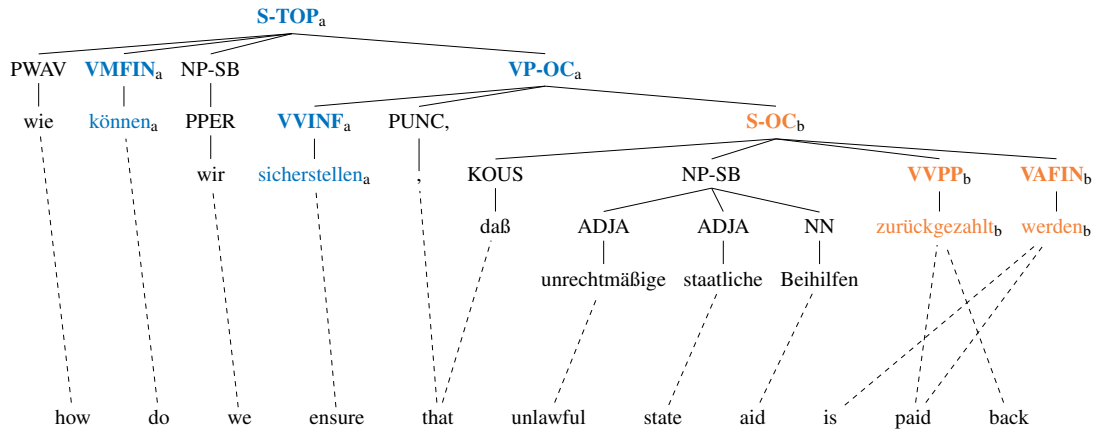


Figure 8.6: An alignment graph for a sentence pair from the training data. The target sentence has two verbal complex sets, indicated by colour (and by the subscripts a and b).

a verbal complex, their lowest common clause node, and any intermediate constituent nodes.

As with agreement and government relations, the Tiger treebank annotation makes it possible to identify verbal complexes using a few syntactic patterns. We use a simple rule-based procedure that identifies verbs and then groups them according to constituency patterns. The procedure is fully described in Appendix C.

Induction of Constraints

As for agreement and government constraints (Section 7.5.2), verbal complex constraint induction is defined for an SCFG grammar rule r in terms of the subgraph h of the alignment graph from which it was extracted. We use the same notation, which we repeat here: if r is written

$$\mathrm{Y}_0 \rightarrow \mathrm{X}_1\ \mathrm{X}_2 \ldots \mathrm{X}_m \ \mid\ \mathrm{Y}_1\ \mathrm{Y}_2 \ldots \mathrm{Y}_n$$

then the head symbol Y_0 is projected from the root node of h and the target body symbols, $\mathrm{Y}_1 \ldots \mathrm{Y}_n$, are projected from its sink nodes. For a rule symbol Y_i we will write h_i to denote the projecting node of the subgraph. If Y_i is a terminal we will write $\mathrm{pos}(i)$ to denote the part-of-speech label from h_i's parent node.

The constraints for each rule r are generated as follows:

1. For each pair i and j, $i < j \leq n$, for which h_i and h_j belong to a common verbal complex set S and where i is the least value such that $h_i \in S$, the following identity

is a constraint: $\text{Y}_i(\text{VC}) = \text{Y}_j(\text{VC})$

2. If $\text{Y}_i(\pi)$ is a constraint term and Y_i is a terminal then $\text{Y}_i(\text{CAT}) = \text{pos}(i)$ is also a constraint.

In the resulting constraint set, every constraint is associated with exactly one verbal complex. This means that the constraint set can be partitioned such that there is one subset of constraints for each verbal complex set.

Example

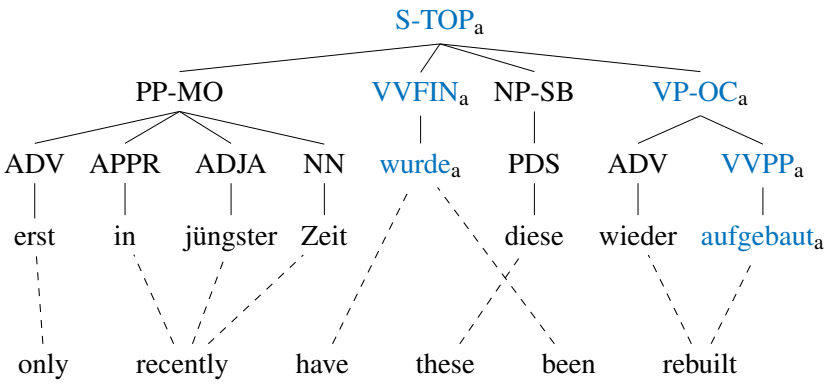


Figure 8.7: An alignment graph for a sentence pair from the training data. The target sentence has a single verbal complex set with node membership indicated in blue (and by the subscript a).

Figure 8.7 shows the annotated alignment graph for a sentence pair from the training data. From this graph, the following two rules could be extracted (among others):

$$\begin{aligned}
&\text{VP-OC} \rightarrow \textit{rebuilt} \mid \textit{wieder aufgebaut} \\
&\qquad \langle \text{VP-OC VC} \rangle = \langle \textit{aufgebaut} \text{ VC} \rangle \\
&\qquad \langle \textit{aufgebaut} \text{ CAT} \rangle = \texttt{VVPP}
\end{aligned}$$

$$\begin{aligned}
&\text{S-TOP} \rightarrow \text{X}_1 \textit{ have } \text{X}_2 \textit{ been } \text{X}_3 \mid \text{PP-MO}_1 \textit{ wurde } \text{NP-SB}_2 \text{ VP-OC}_3 \\
&\qquad \langle \text{S-TOP VC} \rangle = \langle \textit{wurde} \text{ VC} \rangle \\
&\qquad \langle \text{S-TOP VC} \rangle = \langle \text{VP-OC VC} \rangle \\
&\qquad \langle \textit{wurde} \text{ CAT} \rangle = \texttt{VAFIN}
\end{aligned}$$

8.6 Experiments and Analysis

8.6.1 BLEU

We first compare our baseline system from Chapter 6 to systems in which compatibility of verbal complex values is enforced using hard constraints, both with and without the REQUIRE-COMPLETE check. We will refer to these as the "HC" and "HC, complete" systems, respectively. The hard constraint systems are otherwise identical to the baseline (including tuning weights). The constraints produce changes to the 1-best output of between 7.5% and 7.9% of sentences in the dev set and three test sets, but this only results in negligible changes to the BLEU scores (the largest change is +0.1 BLEU).

In the remainder of this section we present a finer-grained analysis to determine what effect our constraints are having and to gain a clearer picture of where the problems lie in verbal complex production.

8.6.2 Accuracy of Verbal Complex Types

In Section 8.2 we gave the following example of an ungrammatical verbal complex from our baseline system: 'hat ... kritisiert worden.' The model we have developed is designed to ensure that translations use grammatical and complete combinations of auxiliary and main verbs. However manually inspecting the changes between the baseline and hard constraint systems indicates that the latter system often satisfies the constraints by producing structures that are grammatically complete but have a different meaning (for example, 'hat ... kritisiert' or just 'hat ... ').

We can distinguish four outcomes of verbal complex production. The verbal complex can be:

1. Grammatical, with the correct feature values (tense, mood, etc.)
2. Grammatical, with the wrong feature values
3. Incomplete
4. Inconsistent

Clearly, there is little benefit to eliminating verbal complexes of types 3 and 4 if they are only being exchanged for ones of type 2. In the rest of this section, we attempt to measure how often the four outcomes occur in our baseline and hard constraint systems. We do this by extracting reference feature structure values from our reference translations and using the standard precision, recall, and F1 metrics.

Reference Feature Structure Values

We extract verbal complex values from our reference sets using the following steps:

1. Parse the reference sentences.
2. Annotate the trees with verbal complex set membership (as for the training data).
3. For each verbal complex, lookup feature structure sets for each verb in the lexicon and unify.

Simple Declarative Sentences

To calculate accuracy metrics across all verbal complex values we would require alignments between the clause nodes of the test and reference trees. Accurately aligning clauses is a challenging task in its own right and so we restrict our analysis to a simpler task: translations of simple declarative sentences. That is, sentences comprising a single finite declarative clause with no finite subordinate clauses. For most such sentences, the translation is also a simple declarative sentence (in the training data, this is true for 99.1% of cases) and we restrict our analysis to those instances. With this simplification, we base our accuracy measures on the count of matches between the top-level verbal complex values in the test and reference sentences.

To select test sentences, we first parse the source-side of the dev and test sets using the Berkeley parser (Petrov and Klein, 2007), which is trained on the Penn Treebank. We filter out sentences that are not simple or declarative, basing our identification criteria on the treebank annotation guidelines (Bies et al., 1995) (a sentence must be an S with an NP child and a finite VP child, in that order, and with no finite subclauses). We then parse the reference translations using the BitPar parser and remove sentences that are not simple and declarative. This leaves 491, 533, 584, and 738 sentences for the 2008, 2009, 2010, and 2011 sets, respectively.

We modify our baseline to use soft constraints with a weight of 0. This allows us to generate a trace of verbal complex feature structures for the baseline. The addition of constraint model state to the baseline affects search, albeit minimally. In order to get a truer representation of the baseline behaviour, we drop any sentences for which the 1-best model score is changed by the introduction of soft constraints.[2]

[2]Over the full 2008, 2009, 2010, and 2011 sets, this was found to amount to 0.9% of sentences. The largest change in BLEU score compared to the constraint-free baseline was for 2008, which (fortuitously) increased from 15.68 to 15.70 due to the change in search behaviour.

Precision, Recall, and F1

Having generated test and reference feature structure values, we count the number of matches and report precision, recall, and F1-measure values, where: if m is the number of matches, t the total number of test set values, and g the number of gold values, then precision p and recall r are:

$$p = \frac{m}{t}, \quad r = \frac{m}{g}$$

and F1 is

$$F1 = \frac{2pr}{p+r}$$

These three measures are widely used in reporting task accuracy in NLP evaluation. For example, in the PARSEVAL metric, which measures labelled bracketing accuracy for parser output (Black et al., 1991).

Our count, t, of test set values excludes empty and incomplete feature structures. An empty feature structure indicates either that unification failed or that there are no VC identities (which should only occur if the clause is without a finite verb). We test for incompleteness by checking for the presence of empty FOUND values.

Table 8.3 shows the results for the baseline and the two hard constraint systems. While the number of incomplete and inconsistent values is reduced by the constraints (indicated by the decreasing $g-t$ values), the numbers of values matching the reference (the m values) are only minimally increased. This leads to slight increases in recall with some loss in precision, although the effects are small.

8.6.3 Error Classification

We have claimed that empty and incomplete feature structures indicate translations containing verbal complexes that are incomplete or inconsistent. Table 8.3 suggests that these account for a significant proportion of verbal complexes (between 10.2% and 12.8% depending on data set). In order to verify that these are genuine translation errors and to understand the types of errors that occur, we manually check 150 sentences from our baseline system and classify the errors. The sample is chosen at random (without replacement) from the 227 sentences in the 2009, 2010, and 2011 baseline output for which the top-clause feature structure is empty or incomplete.

We choose to categorize the errors in terms of German-specific grammatical errors rather than adopt one of the typologies that have been proposed for general SMT error analysis. For describing errors in verbal complexes, general error classifications tend

Data Set	Experiment	t	g	m	$g-t$	Prec.	Recall	F1
2008	Baseline	433	482	214	49	49.4	44.4	46.8
	HC	444	482	221	38	49.8	45.9	47.7
	HC, complete	454	482	224	28	49.3	46.5	47.9
2009	Baseline	458	525	215	67	46.9	41.0	43.7
	HC	474	525	222	51	46.8	42.3	44.4
	HC, complete	481	525	223	44	46.4	42.5	44.3
2010	Baseline	506	574	273	68	54.0	47.6	50.6
	HC	520	574	272	54	52.3	47.4	49.7
	HC, complete	529	574	274	45	51.8	47.7	49.7
2011	Baseline	633	725	323	92	51.0	44.6	47.6
	HC	644	725	328	81	50.9	45.2	45.2
	HC, complete	671	725	335	54	49.9	46.2	48.0

Table 8.3: Feature structure accuracy for the development set (2008) and three test sets (2009, 2010, and 2011). The VC values of the output are compared against the reference (or 'gold') values giving the number of matches (m). The counts t, g, and m (the numbers of test values, gold values, and matches, respectively) are used to compute precision, recall, and F1 values.

to be too broad (for example, all errors would be classed simply as 'syntactic' under the typology of Farrús et al. (2010)) or they tend not to be particularly illuminating for this particular task (for example, "missing words / filler," "word order / word level" in the typology of Vilar et al. (2006)).

For each error category we give the percentage of errors found:

Inconsistent Combination (36.0%) An ungrammatical combination of auxiliary and main verbs.
Example: *zwei Häftlinge* ***haben*** *nicht* ***überleben*** .

Perfect missing aux (21.3%) There is a past-participle in sentence-final position, but no auxiliary verb.
Example: *die Eltern 1600 Kronen oder mehr pro Impfung* ***bezahlt*** .

No verb (11.3%) The input contains at least one verb that should be translated but the output contains none.
Example: *Fluocompact Lampen im Durchschnitt zwischen 6000 und 10.000 Stunden* .

False positive (10.7%) The verbal complex is grammatical. In the sample this is either because the output string is well-formed in terms of verb structure, but the tree is wrong, or the parse of the source is wrong and the input does not actually contain a verb.

Invalid sentence structure (10.0%) The verbs are present and make sense, but the sentence structure is wrong.
Example: *der Ausschuß Gélineau in Quebec , das Mandat* ***erfüllt hat*** .

Infinitive missing auxiliary / misplaced finite verb (6.7%) There is an infinitive in sentence-final position, but no auxiliary verb *or* the main verb is erroneously in final position (the output is likely to be ambiguous).
Example: *Glücklicherweise Tabellen der Berechnungen* ***erleichtern*** .

Unknown verb (2.7%) The source verb is untranslated.
Example: *dann* ***scurried*** *ich zu meinem Sitz* .

Werden-passive missing aux (1.3%) There is a *werden*-passive non-finite part, but no finite auxiliary verb.
Example: *" Aber keine konkreten Abschluss* ***erzielt worden*** .

In our classification, the most common individual error type in the baseline is the inconsistent combination of verbs, at 36.0% (54 out of 150). However, there are multiple categories that can be characterized as the absence of a required verb: the "perfect missing aux," "no verb," and "werden-passive missing aux" categories all involve incomplete verbal complexes. The category "infinitive missing auxiliary / misplaced finite verb" is an ambiguous error type that may or may not indicate a missing auxiliary. Combined, these incomplete verbal complex categories total between 33.9% and 40.6% of the errors, depending on whether errors of the latter category are included. There are also some false negatives (10.7%) and potentially misleading results in which wider syntactic errors result in the failure to produce a feature structure (10.0%), but the majority are genuine errors.

8.7 Conclusion

In this chapter we have investigated the task of verbal complex translation. We developed a lexicon-based representation of verbal complex types as feature structure values and added constraints to enforce consistency of the values produced during clause composition.

By extracting verbal complex values from the reference sentences, we measured accuracy in our baseline and hard constraint systems using standard precision, recall, and F1 metrics. We found that the baseline failed to produce complete and consistent feature structures for between 10.2% and 12.8% of simple declarative sentences (depending on data set) and performed a detailed analysis to verify that these resulted from genuine translation errors. We found that whilst there were some false positives resulting from wider syntactic errors, the majority of empty or incomplete feature structure values were the result of errors in verbal complex production, indicating that our method can successfully detect errors during clause composition. Our hard constraint systems performed better than the baseline in terms of producing complete and consistent verbal complexes and in terms of recall, but the improvement was small.

We conclude from our analysis that removing ill-formed hypotheses from the search is insufficient if the model frequently allows source verbal complexes to be translated as incomplete target verbal complexes, not translated at all, or translated with the wrong grammatical features. This finding motivates the work in the next chapter where we use additional information derived from the source-side to influence the production of verbal complexes on the target-side.

Chapter 9

Improving Verbal Complex Production

9.1 Introduction

In the previous chapter, we developed a representation of German verbal complexes as feature structures and used constraints to ensure that consistent and complete values were produced during translation. However, we found that using constraints alone led to little improvement over the baseline. Our analysis indicated that the model did little better than the baseline at producing verbal complexes with the correct grammatical features (as compared to the reference) and that many source verbal complexes were allowed to go untranslated.

In this chapter, we extend our verbal complex model to use information about clause structure and source verbs derived from a parse of the input sentence. We develop two feature functions: one that penalises the non-translation of source verbal complexes and one that scores a target verbal complex choice based on the content of the source verbal complex. In experiments, we repeat the precision and recall measurements of the last chapter and find that using our feature functions leads to improvements over our earlier results.

9.2 Model

We add two verbal-complex-specific feature functions to our baseline model. In order to define the feature functions, we first describe 'clause projection,' a simple source-syntactic restriction on decoding. We then describe our heuristic method of obtaining probability estimates for a target verbal complex value given the input clause.

Experiment	2008	2009	2010	2011
Baseline	15.7	15.0	16.6	15.4
Clause projection	15.8	15.1	16.9	15.5

Table 9.1: BLEU scores for the development and test sets with and without clause projection.

9.2.1 Clause Projection

In order to define our feature functions we require that we have an alignment from source-side clauses to target clauses. However, our translation model does not use source syntax and, as was discussed in Section 2.5.4, naively adding it to string-to-tree models has been found to perform poorly without the development of more complex translation models.

Rather than incorporate full syntactic structure on the source-side, we adopt a simple restriction that finite declarative clauses (both main and embedded) on the source-side must be translated as clauses on the target-side. This is clearly an over-simplification from a linguistic perspective but it appears not to harm translation quality in practice. Table 9.1 shows BLEU scores for our baseline system run with and without this restriction.

It is perhaps not too surprising to find that this selective introduction of source syntax can improve translation: Marton and Resnik (2008), working with a Chinese-English hierachical phrase-based model, found that when grammar rules were required to match the bracketing of a source parse tree during decoding, the benefit or harm of imposing such constraints was highly dependent on constituent type. For example, using a penalty feature for rules that violate the bracketing of Chinese source PPs proved harmful, whereas a similar penalty was beneficial for NPs. Whilst we leave a deeper investigation of clause projection to future work, it seems reasonable to assume that, on aggregate, the benefit of using additional context (from the source parse) for rule selection outweighs the harm of disallowing legitimate structural divergence between English and German clause structures.

Clause projection is implemented as follows:

1. The input sentence is parsed and a set of clause spans is extracted according to the 1-best parse. We use the Berkeley parser (Petrov and Klein, 2007), which is trained on the Penn Treebank and so we base our definition of a declarative

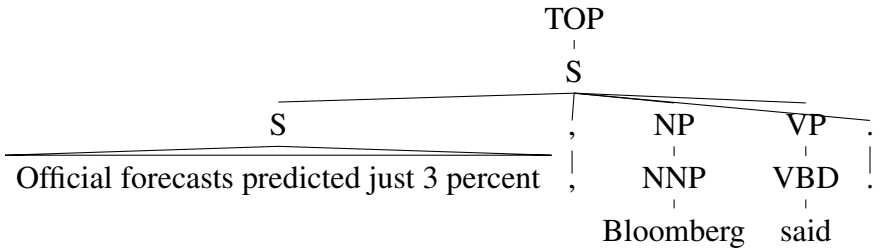
clause on the treebank annotation guidelines.

2. The clause spans are adjusted to take account of the differing attachment styles for sentence-ending punctuation between the Penn and Tiger treebanks (in the Penn Treebank the punctuation is a child of the clause node whereas in Tiger it is a sibling, with the clause and punctuation nodes being children of the TOP node).

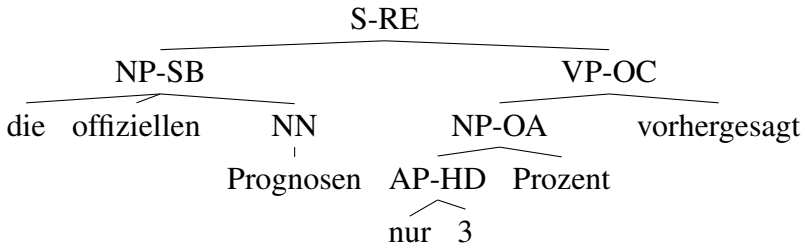
3. We modify the decoding algorithm to produce derivations in chart cells only if the cell span is consistent with the set of clause spans (i.e. if source span [i,j] is a clause span then no derivation is built over span [m,n] where $i < m \leq j$ and $n > j$, etc.).

4. We modify the decoding algorithm so that grammar rules can only be applied over clause spans if they have a clause label ('S' or 'CS', since the parser we use is trained on the Tiger treebank).

As an example, consider the following input sentence, which uses an embedded clause to report indirect speech:



Under our model of clause alignment, the decoder is forced to build clause nodes over source spans [1,6] and [1,9]. For this sentence, the 1-best derivation already does so, with the following sub-derivation for span [1,6]:



However, the translation for this embedded clause contains an incomplete verbal complex resulting from the application of a grammar rule containing a clause with no finite

verb. The source verb 'predicted' *is* translated, but it is translated within the child VP-OC to a past-participle or adjective ('vorhergesagt'). With a model of clause alignment, we are able to define feature functions that use properties of an input clause to influence the verbal complex produced in the corresponding target clause.

9.2.2 Verbal Complex Probabilities

When translating a clause, the source-side verbal complex will often provide sufficient information to select a reasonable type for the target verbal complex, or to give preferences to a few candidates.

By matching up source-side and target-side verbal complexes we estimate co-occurrence frequencies in the training data. To do this for all pairs in the training data, we would need to align clauses between the source and target training sentences. However, it is not crucial that we identify every last verbal complex. We simplify the task by restricting training data to sentence pairs in which both source and target sentences are declarative sentences, making the assumption that the main clause of the source sentence aligns with the main clause of the target.

We represent source-side verbal complexes with a label that is the string of verbs and particles and their POS tags in the order that they occur in the clause, for example, `plays_VBZ` and `is_addressing_VBZ_VBG`. The target-side feature structures are generated by identifying verbal complex nodes in the training data parse trees (as in Section 8.5.2) and then unifying the corresponding feature structures from the lexicon.

Many source verbal complex labels exhibit a strong co-occurrence preference for a particular target type. For example, Table 9.2 shows the three most frequent feature structure values for the target-side clause when the source label is `is_closed_VBZ_VBN`. The most frequent value corresponds to a non-modal, *sein*-passive construction in the present tense and indicative mood.

9.2.3 Feature Functions

We add two feature functions to the baseline model, h_{vcm} and h_{mvp} (where *vcm* stands for "verbal complex model" and *mvp* stands for "missing verb penalty"). As with the baseline feature functions, our verbal complex-specific functions are evaluated for every rule application r_i of the synchronous derivation. Like the language model feature function, they are non-local and so cannot be pre-computed. Unlike the baseline functions, the value returned depends on whether the source span that the rule is applied to

RF	F-Structure
0.841	[FIN [CAT AUX; LEMMA sein; MOOD indicative; TENSE present]; NON-FIN [CAT PP/SP; PP [LEMMA *]]]
0.045	[FIN [CAT FULL; LEMMA sein]; NON-FIN none]
0.034	[FIN [CAT AUX; LEMMA werden; MOOD indicative; TENSE present]; NON-FIN [CAT WPP; PP [LEMMA *]; WERDEN none; WORDEN none; SEIN none]]
...	...

Table 9.2: Observed values and relative frequencies (RF) for *is closed*, which was observed 44 times in the training data. For readability, none-values have been omitted for top-level features.

is a declarative clause or not.

Both feature functions are defined in terms of X, the frontier VC feature structure value of the sub-derivation at rule application r_i. If there are multiple frontier VC values then the scores are calculated for each value and the maximum is taken. If there are no frontier VC values then the score is calculated as if X were the empty value, $[]$.

The first feature function, h_{vcm}, uses the source verbal complex label, l, and the relative frequency probability estimate, $P(X|l)$, learned from the training data:

$$h_{vcm}(r_i) = \begin{cases} P(X|l) & \text{if } r_i \text{ covers a clause span with source} \\ & \quad \text{verbal complex label } l \text{ and } c_l \geq c_{min} \\ 1 & \text{otherwise} \end{cases}$$

The probability estimates are not used for scoring if the number of training observations falls below a threshold, c_{min}. We use a threshold of 10 in experiments.

The second feature function, h_{mvp}, is simpler: it penalizes the absence of a non-empty verbal complex value when translating a source declarative clause:

$$h_{mvp}(r_i) = \begin{cases} exp(1) & \text{if } r_i \text{ covers a clause span and } X \text{ is empty} \\ 1 & \text{otherwise} \end{cases}$$

Unlike h_{vcm}, which requires the source verbal complex label to have been observed a number of times during training, h_{mvp} is applied to all source spans that cover a declarative clause.

Dropped verbs were found to be a frequent problem in our baseline model (Section 8.6.3) and this function, together with the REQUIRE-COMPLETE check from the previous chapter, is intended to curb this tendency.

9.3 Experiments and Analysis

In this section we compare our baseline system against five systems:

HC, complete	This is identical to the hard constraint system from the previous chapter (with the REQUIRE-COMPLETE check).
HC, complete, CP	As "HC, complete" but also uses clause projection.
h_{mvp}	As "HC, complete, CP" but also uses the missing verb penalty feature function.
h_{vcm}	As "HC, complete, CP" but also uses the verbal complex model feature function.
$h_{mvp} + h_{vcm}$	As "HC, complete, CP" but uses both feature functions.

We found BLEU proved too coarse-grained to measure changes in verbal complex accuracy, so we instead tuned the weights of the h_{mvp} and h_{vcm} features by running a line search to optimize the F1 score (as defined in Section 8.6.2) on a subset of the news-test2008 development set containing sentences up to 30 tokens in length. We first tuned the weight for h_{vcm} and then tuned h_{mvp} with the h_{vcm} weight fixed.

We first present BLEU scores and then return to verbal complex accuracy.

9.3.1 BLEU

Table 9.3 shows BLEU scores for the baseline and five test systems. Preliminary experiments showed that re-tuning the system weights made little difference to the BLEU scores and so we re-use the tuning weights for the baseline feature functions to avoid introducing variance from the weight optimisation process. There is a small gain from using clause projection, but no additional gain in BLEU using our two feature functions.

Experiment	2008	2009	2010	2011
Baseline	15.7	15.0	16.6	15.4
HC, complete	15.7	15.0	16.6	15.4
HC, complete, CP	15.8	15.1	16.9	15.6
h_{mvp}	15.8	15.2	16.8	15.6
h_{vcm}	15.7	15.1	16.9	15.6
$h_{mvp} + h_{vcm}$	15.8	15.1	16.8	15.6

Table 9.3: BLEU scores for the development and test sets with clause projection (CP) and the two verbal complex feature functions, h_{mvp} and h_{vcm}.

9.3.2 Feature Structure Accuracy

Table 9.4 shows accuracy results for the baseline and the five constraint systems, calculated as in the previous chapter (Section 8.6.2). The results for the three test sets are similar and so we sum their t, g, and m counts and give aggregate precision, recall, and F1 values.

The h_{mvp} and h_{vcm} feature functions appear to be effective at reducing the number of incomplete and inconsistent values (the $g - t$ column), with the lowest values being achieved by a combination of the two. In the previous chapter we found that using constraints alone led to slight increases in recall with some loss in precision for the test sets, whereas we now achieve increases in both precision and recall.

Whilst combining h_{mvp} and h_{vcm} produces the highest counts of complete VC values, this comes at a slight loss in precision over using h_{vcm} alone.

Data Set	Experiment	t	g	m	$g-t$	Prec.	Recall	F1
dev	Baseline	433	482	214	49	49.4	44.4	46.8
	HC, complete	454	482	224	28	49.3	46.5	47.9
	HC, complete, CP	442	482	223	40	50.5	46.3	48.3
	h_{mvp}	460	482	228	22	49.6	47.3	48.4
	h_{vcm}	470	482	251	12	53.4	52.1	52.7
	$h_{mvp} + h_{vcm}$	475	482	251	7	52.8	52.1	52.5
test	Baseline	1,597	1,824	811	227	50.8	44.5	47.4
	HC, complete	1,681	1,824	832	143	49.5	45.6	47.5
	HC, complete, CP	1,663	1,824	835	161	50.2	45.8	47.9
	h_{mvp}	1,713	1,824	857	111	50.0	47.0	48.5
	h_{vcm}	1,764	1,824	941	60	53.3	51.6	52.5
	$h_{mvp} + h_{vcm}$	1,777	1,824	941	47	53.0	51.6	52.3

Table 9.4: Feature structure accuracy for the development set and three test sets (aggregated). As in Table 8.3, the vc values of the output are compared against the reference (or 'gold') values giving the number of matches (m). The counts t, g, and m (the numbers of test values, gold values, and matches, respectively) are used to compute precision, recall, and F1 values.

9.3.3 Manual Analysis

In order to get a clearer picture of the changes produced by our features, we perform a manual analysis of a sample of 100 sentences. We select sentences for which, according to our metric, the baseline feature structure does not match the reference, but the h_{vcm} feature structure does. We choose the h_{vcm} system for comparison because it achieves the highest F1 scores.

We chose 100 sentences at random (without replacement) from the three test sets. The sentences are categorised according to whether the comparison against the reference is correct (Y) or not (N) for the baseline and h_{vcm} system respectively. For each category we give the number of sentences in parentheses:

Y/Y (69)

Verb constructions are grammatical and we agree with both comparisons against the reference value: the baseline system does not match the reference value but the h_{vcm} system does.

In the following example, the baseline output does not contain a verb, whereas the h_{vcm} output does:

Input	His step-daughter **went** to Plymouth .
Ref.	Seine Stieftochter **ging** nach Plymouth.
Baseline	Seine step-daughter nach Plymouth .
h_{vcm}	Seine step-daughter **ging** nach Plymouth .

In the next example, the ungrammatical verbal complex “hatte . . . montiert worden” is replaced by “war . . . versammelt,” which is grammatical and matches the reference feature structure type.

Input	The commission **had been assembled** at the request of Minister of Sport Miroslav Drzeviecki .
Ref.	Die Kommission **war** auf Anfrage von Sportminister Miroslaw Drzewiecki **zusammengekommen** .
Baseline	Die Kommission **hatte** auf Antrag der Minister für Sport Miroslav Drzeviecki **montiert worden** .
h_{vcm}	Die Kommission **war** auf Antrag der Minister der Sport Miroslav Drzeviecki **versammelt** .

N/Y (18)

Verb constructions are grammatical. We agree with the comparison for the test system but not the baseline.

In the following example, the verbal complex is of the same type in the baseline as the reference, but the verb, *erleichtert*, appears in the wrong position leading to its misinterpretation as an infinitive with a missing auxiliary rather than as a finite verb. Whilst the metric classifies this as an error, this is due to a problem with sentence structure rather than verb choice.

Input	The Catalan group **facilitated** thousands of passports to Al Qaida
Ref.	Die katalanische Zelle **besorgte** Al Qaida tausende von Reisepässen
Baseline	Die katalanische Gruppe Al Qaida Tausende von Pässen **erleichtert**
h_{vcm}	Die katalanische Gruppe **erleichtert** Tausende von Pässen zu Al Qaida

Y/N (13)

We agree with the comparison for the baseline output but not h_{vcm}.

In the following example, the baseline and h_{vcm} output both begin *Côte d ' Ivoire ist nervös* ("Ivory Coast is nervous") and then insert a translation of "expecting" somewhere later in the sentence. The reference simply uses *erwartet* ("expects").

Input	Ivory Coast **is** nervously **expecting** the announcement of the election results .
Ref.	Elfenbeinküste **erwartet** aufgeregt die Ankündigung der Wahlergebnisse
Baseline	Côte d ' Ivoire **ist** nervös , die Bekanntgabe der Wahlergebnisse **erwartet** .
h_{vcm}	Côte d ' Ivoire **ist** nervös , **erwarten** die Bekanntgabe der Wahlergebnisse .

In the baseline derivation, *Côte d ' Ivoire ist nervös* is the main clause, with a subordinate clause following the comma, and so the verb is taken to be *ist*. In the h_{vcm} derivation, *Côte d ' Ivoire ist nervös* is a subclause and so the verb is taken to be *erwarten*. The verb *ist* belongs to a separate verbal complex feature structure.

9.4 Conclusion

In this chapter we extended our verbal complex model to incorporate source-side information. We used source syntax to identify declarative clause spans and we used feature functions to encourage the accurate production of verbal complex types in the corresponding target clauses.

In the previous chapter we found that our hard constraint systems improved recall at the expense of precision. With the addition of our feature functions, we were able to achieve increases in both precision and recall while substantially reducing the number of clauses for which an empty or incomplete verbal complex was produced (from around 12.4% of single-clause declarative test sentence for the baseline to 2.6% for the $h_{mvp} + h_{vcm}$ system). This adds weight to our finding that removing ill-formed hypotheses from the search may prove an insufficient strategy if the model does not also have some means of encouraging well-formed hypotheses.

Chapter 10

Conclusions and Future Work

10.1 Conclusions

Morphology and syntax have both received attention in statistical machine translation research, but they are usually treated independently and the historical emphasis on translation into English has meant that many morphosyntactic issues remain under-researched. In computational and theoretical linguistics, feature structures and unification have proven to be powerful tools for modelling many aspects of morphosyntax and in this thesis we proposed a framework for extending string-to-tree statistical machine translation models by adding a lexicon of feature structures and adding unification-based constraints to the target-side of the synchronous grammar.

To demonstrate the viability and effectiveness of our proposed approach, we extended a full scale, state-of-the-art, string-to-tree baseline, adding constraints designed to address two quite dissimilar aspects of morphosyntax, both of which are prominent sources of error in German translation output. For agreement and government, which are the linguistic phenomena underlying much of surface form inflection in Indo-European languages, we used the lexicon to link surface form target words with the grammatical features values that are relevant to inflection and we used constraints to model the relations between selectors (controllers and governers) and their targets. We were able to improve translation quality by up to 0.5 BLEU over a strong baseline model. Manual evaluation verified that our constraints led to an overal improvement in translation quality (with the two annotators preferring 2.0 and 2.6 sentences to every one they dispreferred). We found that hard constraints outperformed soft constraints, contrary to our previous finding in Williams and Koehn (2011). We attribute this mainly to improvements in training since that work.

For verbal complex production, which in German can involve wide discontinuities, we represented verbal complex types as feature structure values and used constraints to ensure that the combination of main and auxiliary verbs was both consistent and complete. We extracted reference values from parses of the reference set and used these to measure accuracy. We found that constraints alone were inadequate since the decoder was still able to drop verbs, but by using source-side information about declarative clauses and their verbs we were able to improve accuracy.

To summarise the contributions of this thesis:

- We have presented a language-independent framework for incorporating additional linguistic information into syntax-based translation models using the well-understood machinery of feature structures and unification.

- We described a means of efficiently integrating constraint evaluation into parsing-based decoding. We demonstrated that, although there are non-negligible computational costs, our approach is viable for full-scale translation tasks, provided that feature structures and constraints are carefully designed. For agreement and goverment constraints, decoding time increased by 16.7% over the baseline but the empirical exponent for sentence length versus decoding time did not increase.

- We have developed models for agreement, government, and verbal complex formation in German, demonstrating improvements in translation quality over a strong baseline model.

- We have demonstrated that constraints can be useful for pinpointing translation errors in a system. In our manual analysis in Chapter 8, we found that approximately 80% of incomplete or inconsistent feature structures indicated genuine translation errors. Using this semi-automatic approach, we presented a fine-grained error analysis for verbal complex formation in the baseline.

10.2 Future Work

In this final section we suggest several directions for future work. We first consider the models of English-German that were developed in Chapters 7, 8, and 9: in Section 10.2.1, we highlight the aspects of those models where we feel that further research and refinement is most likely to lead to improvements in translation quality beyond the current level; and in Section 10.2.2, we suggest some extensions that fall within the

scope of the current models. Next, in Section 10.2.3, we describe how similar models could be developed for target languages other than German; finally, in Section 10.2.4, we suggest some other linguistic phenomena that could be modelled using the framework proposed in this thesis.

10.2.1 Improvements to the Current Models

Additional Feature Functions for the Soft Constraint Model

In the present work, soft constraints have been implemented using a single feature (a count of the number of constraint failures). In future work, it would be worth exploring whether a richer set of features could lead to a better soft constraint model.

Since the accuracy of our constraint extraction method is dependent on parse accuracy, one direction for developing a richer feature set would be to learn features that use characteristics of the parse trees to predict constraint reliablity. For instance, it seems plausible that constraints learned from short sentences will be more reliable than those learned from long sentences. Similarly, constraints learned from common syntactic structures may be more reliable than those learned from obscure constructions. Features derived from a subtree might indicate properties like constituent type, head words, tree depth, numbers of nodes, and so on.

By using large-scale discriminative training methods such as MIRA (Chiang et al., 2009) or PRO (Hopkins and May, 2011) researchers have shown that it is possible to learn the parameters of models with large and highly-specific feature sets. Chiang et al. (2009) demonstrate that large improvements in translation quality are possible by using well-chosen syntactic features.

Improving Search for the Hard Constraint Model

When using hard constraints, constraint evaluation is performed immediately prior to the beam-filling step (Section 5.3.3). Constraint failure does not affect the order of cube exploration. It should be possible to improve search accuracy by integrating cube pruning and constraint evaluation.

As an example, suppose that an input sentence contains the substring “a major recession could break this cycle.” One promising rule for that span might be the following:

$$\text{S-TOP} \rightarrow a\ \text{X}_1\ \textit{recession}\ \text{X}_2\ \text{X}_3 \mid \textit{eine}\ \text{ADJA}_1\ \textit{Rezession}\ \text{VMFIN}_2\ \text{VP-OC}_3$$
$$\langle\ \textit{eine}\ \text{AGR}\rangle = \langle\ \text{ADJA}\ \text{AGR}\ \rangle$$
$$\langle\ \textit{eine}\ \text{AGR}\rangle = \langle\ \textit{Rezession}\ \text{AGR}\ \rangle$$

The highest scoring hypotheses in the ADJA stack might look something like the following (the scores were taken from a baseline system during development):

Target words	Score
großen	-0.404921
große	-0.465708
großes	-0.539664
bedeutenden	-0.583893
wichtigen	-0.584497
wichtige	-0.600739
bedeutende	-0.606069
großer	-0.607750
größeren	-0.619415
bedeutender	-0.629519

Target words that are grammatical in the context of the phrase *eine* ... *Rezession* are indicated in bold.

For lexicalised rules like the example, search could be improved by applying constraints prior to cube pruning in order to pre-filter the stack dimensions. For the example, this would filter the ADJA stack to leave the following entries:

Target words	Score
große	-0.404921
wichtige	-0.600739
bedeutende	-0.606069
erhebliche	-0.677803
größere	-0.680824
wesentliche	-0.695622
umfangreiche	-0.788670
schwere	-0.823251
größte	-0.845443
maßgebliche	-0.850760

When a SCFG rule has multiple non-terminals, the search over possible hypotheses is likely to be quite shallow due to the tight bounds on cube pruning (the default setting in Moses generates 1,000 hypotheses per cell).

A preliminary implementation has shown improvements in model score, but only a very modest increase in BLEU score (roughly 0.05 using the baseline weights). Our initial implementation was slow, but we intend to address the efficiency issues in future work and to continue to investigate alternative search strategies.

Improving the Verbal Complex Model

There are several aspects of the verbal complex model that may be worth examining further in future work:

- Clause projection relies on the 1-best parse containing accurate clause structure information. In tree-to-string models, it has been found that using the k-best input trees or a parse forest instead of the 1-best parse leads to improvements (Mi and Huang, 2008). Taking multiple possible clause structures into account would reduce the impact of parse errors.

- Clause projection is a considerable linguistic oversimplification. A better model for mapping English to German clauses may lead to improvements. One approach would be to train a classifier, perhaps along similar lines to Roark et al. (2012), who use classifiers to predict constituent boundaries in monolingual parsing.

- When estimating verbal complex probabilities, our relative-frequency estimates could be smoothed (for example, with Good-Turing smoothing). For rare or unseen source verbal complexes, the model could back-off to POS-based labels.

10.2.2 Extensions to the Current Models

Incorporating Source-Side Information for Agreement and Government

In Chapter 9, we extended our verbal complex model finding that by incorporating additional source-side information we could influence feature structure production leading to a better match with the reference. Whereas English verbal complexes express a similar range of grammatical feature values to German ones, this is less true for English and German inflection. For instance, we would find scant information on the

source-side for influencing the selection of German gender values on the target. However, there are still important aspects of German inflection that should be predictable from the English source:

- It may be possible to improve case prediction by using syntactic cues in the input, perhaps by taking a similar approach to Avramidis and Koehn (2008), who use phrase-structure parse trees to mark English NPs with Greek-like case information for English to Greek factored phrase-based translation. Where they use factors to map case, we would use constraints.

- Number syncretism is far more common for German nouns than English (where examples, such as "sheep", are very rare). For example, in the genitive case, the surface form *Handys* ("mobile phone") is used for both the singular and plural. When translating "mobile phone" or "mobile phones" to *Handys* the number distinction is lost. By using a morphological analyser to determine source-side noun number in the training data, we could add target-side constraints that specify target the number. For example,

$$\begin{aligned} \text{NN} &\rightarrow \textit{mobile phone} \mid \textit{Handys} \\ \langle \textit{Handys}\ \text{INFL AGR NUM} \rangle &= \texttt{sg} \end{aligned}$$

  There are a few nouns that are singular in English but plural in German, and vice versa. For instance, *Polizei* ("police") is singular in German. These could be handled with a stop-list or by using probabilistic constraints.

Pronoun-Antecedent Agreement

In German, as in English, relative pronouns agree with their antecedents. For example, in *Der Mann der die Frau liebt* ("the man who the women loves") the relative pronoun *der* agrees in case, number, and gender with the antecedent, *der Mann*. This could be modelled similarly to subject-verb agreement (Section 7.4.3) by using the lexicon to specify the agreement values of relative pronouns and using constraints to ensure that the antecedent and the relative pronoun have compatible agreement values.

Combining the Constraint Models

In Chapters 7, 8, and 9, we developed two independent constraint models, one for agreement and government, and one for verbal complex production. There is no reason

System	2009-20	2010-20	2011-20
Baseline	16.8	17.1	14.5
+noun-modifier	17.0	17.5	14.7
+prep-gov	17.0	17.5	14.7
+adj-decl	17.0	17.6	14.7
+np-case	17.1	17.8	14.9
+subj-verb	17.2	17.8	14.9
+h_{vcm}	17.3	18.1	15.0

Table 10.1: BLEU scores for short sentences as constraint types are progressively included.

that these constraint models cannot both be used in a single system and we have built a proof-of-concept system that does so, although we have not yet conducted any detailed analysis.

In our combined model, we use the original lexicon and constraint extraction processes unchanged, running them once for each for the two models. Every constraint is assigned an index according to which constraint model it belongs to. During decoding, constraints are evaluated for the two model types in turn, resulting in two independent frontier feature structure sets for each hypothesis. The decoder looks up feature structure values in the constraints' respective lexicons.

Table 10.1 shows the experiment on short sentences from Section 7.6.1 continued to include the h_{vcm} feature function from Chapter 9. Compared to a system with h_{vcm} only, the F1 score is slightly lower at 0.482 compared to 0.487 (the baseline on this test set is 0.423). We suspect that the increase in BLEU is mainly attributable to clause projection.

10.2.3 Application to Other Target Languages

Agreement and Government

Many Indo-European languages have inflectional morphologies as rich as German's, or more so, and cross-linguistically, patterns of agreement and government tend to be similar. Our model and training scheme for agreement and government should be straightforward to apply to other languages provided that suitable language-processing tools are available: namely, a phrase-structure parser and morphological analyser. Our tree

annotation procedure and constraint induction scheme (Section 7.5.2 and Appendix B) were tailored to suit the German language and the conventions of the Tiger Treebank, but similar annotation procedures could be developed for other languages.

Morphological analysers are available for a number of languages with rich inflection, including Russian, Czech, and Hindi. Where treebanks and parsers exist, they tend to use dependency structure (partly due to the freer word order that comes with grammatical case). Formally, conversion to phrase structure is always possible, though non-projective structures require projectivization, which can lead to a loss of information (see Nivre and Nilsson (2005) for a discussion of graph transformation techniques for projectivization). Non-projective dependencies are frequent in some languages: for Czech, Hajičová et al. (2004) report that 2% of words and 19% of sentences have non-projective dependencies. Assuming a method for projectivization, the main question is whether the syntactic content is suited to the task (see Rambow (2010) for a good discussion of this topic). There has been success converting dependency representations to phrase-structure for adapting parsing models (for example, Collins et al. (1999); Xia and Palmer (2001) do so for Czech) and so for languages that have dependency treebanks and parsers, like Czech and Russian, it is likely that similar models could be developed.

Verbal Complex Production

Like German, Dutch also has grammatical constructions in which auxiliary and main verbs are separated by arbitrarily many constituents, but cross-linguistically, such constructions are rarer than long-range agreement and government phenomena. Whilst discontinuities of this sort are especially challenging for statistical machine translation models, they are far from the only problem in verbal complex formation. For example, when we saw 'has been criticized ...' translated to *hat ... kritisiert worden* by our baseline system (Section 8.2), we noted that the problem was not only the choice of auxiliary finite verb, but also the word-by-word verb translation that translated each English verb to a German verb instead of producing the less-literal alternative *wurde ... kritisiert*. For capturing these types of translational preferences, our model is applicable for any target language that uses separate main and auxiliary verbs.

Apart from auxiliary-main verb constructions, there are other similar phenomena that could be modelled using our framework. For instance, many English multi-word verbs are separable ('take off,' 'mark down,' etc.) and constraints similar to those in Chapter 8 could be used to penalize the production of particles without verbs.

Automatically Learning Constraint Annotation Rules

The German models presented in this thesis used manually-developed constraint annotation rules that are, at least to an extent, language- and treebank-specific. To make the application to other languages and treebanks easier, it may be worth exploring unsupervised or semi-supervised methods for constraint learning.

It may be possible to learn tree annotation rules by generating candidate relations according to linguistically-motivated 'universal' principles and then refining the relations using a data-driven approach. For instance, Generalized Phrase-Structure Grammar proposes three universal feature instantiation principles that together form a theory for agreement (Zwicky, 1986). Given a lexicon, candidate relations for the training sentences could be tested and then removed if the words are found not to agree — that is, if unification of agreement values fails. Since fortuitous unification is possible (if the words are not in an agreement relation but happen to have compatible grammatical features), it may be possible to bolster this test with a machine-learning based approach that classifies relations as agreement relations or not according to additional discriminative features, such as part-of-speech *m*-gram sequences, the surface forms of the candidates, and so on.

10.2.4 Modelling Other Linguistic Phenomena

Verb Subcategorization and Verbal Case Government

In unification-based grammars it is common to specify verb subcategorization requirements in the lexicon. For instance, requiring that the intransitive verb 'dream' never takes an argument and permitting the verb 'give' to be used with one, two, or none.

In German, a verb's subcategorization frame usually specifies the cases of the arguments since verbs govern case in German. For example, the verbs *unterstützten* ("to assist") and *helfen* ("to help") both require a single argument, but the argument must be in the accusative case for *unterstützten* and the dative for *helfen*.

A minimal implementation in our framework could use atomic values to specify subcategorization types in the lexicon and in constraints. For example, `acc` might specify that the verb requires a single accusative argument and `dat+acc` might specify that it requires a dative and accusative, in that order.

Given the following lexical entries for the past-tense 3-sg forms of *unterstützten* and *helfen*:

$$\textit{unterstützte} \rightarrow \begin{bmatrix} \text{CAT} & \texttt{VVFIN} \\ \text{SUBCAT} & \texttt{acc} \end{bmatrix} \qquad \textit{half} \rightarrow \begin{bmatrix} \text{CAT} & \texttt{VVFIN} \\ \text{SUBCAT} & \texttt{dat} \end{bmatrix}$$

the rule:

$$\text{S-TOP} \rightarrow \textit{she}\ \text{X}_1\ \text{X}_2 \mid \textit{sie}\ \text{VVFIN}_1\ \text{NP-OA}_2$$
$$\langle \text{VVFIN SUBCAT} \rangle = \texttt{acc}$$

could be used in combination with the verb *unterstützte* but not *half*. For example, it could be used in translating "she helped the man" to *sie unterstützte den Mann* but not *sie half den Mann*, which is ungrammatical since the argument should be in the dative case — *sie half dem Mann* — and requires an alternative rule that specifies a dative argument (NP-DA instead of NP-OA).

An implementation would require a list of possible subcategorization frames for each verb. Weller et al. (2013) describe a method for extracting this information from dependency-parsed German text.

Semantic Relations

As well as morphosyntactic information, feature structures have been used for phonological and semantic properties of words (for instance, in Functional Grammar (Kay, 1979) and Head-Driven Phrase Structure Grammar (Pollard and Sag, 1994)). For machine translation, it may be useful to use the lexicon to restrict the semantic relations in which specific verbs and their arguments can participate. For example, to allow "house" to occur as the object of a clause in which it is "constructed," but not one in which it is "forged."

Given a source of semantic relations, constraints could be used to restrict the combination of verbs and arguments by requiring the presence in the lexicon of matching relation types. Lewis and Steedman (2013) show how distributional clustering can be used to learn relations suitable for use in wide-coverage settings.

Appendix A

Mapping Part-of-Speech Values from Morphisto to Tiger

This appendix describes the mapping from Morphisto to Tiger part-of-speech tags that we use in Chapter 7. The categories used by the morphological analyser are coarser-grained than in Tiger, but finer-grained distinctions are instead encoded using feature values. For example, Morphisto's V category is used for all verbs, whereas Tiger subdivides categories depending on whether i) the verb is an infinitive, a past-participle, finite, or imperative, and ii) it is an auxiliary, modal, or full verb. The morphological analyser makes the former distinction using feature values (such as `<inf>` to indicate an infinitive). The mapping therefore involves the Morphisto part-of-speech value and also the presence or absence of feature tags. For instance, if the Morphisto part-of-speech value is DEM and the analysis contains the `<subst>` feature tag then the Tiger tag is PDS. Table A.1 shows the full mapping scheme.

Morphisto POS	**Morphisto Features**	**Tiger POS**
ADJ or ORD	<Adv> or <Pred>	ADJD
	No <Adv> and no <Pred>	ADJA
ART		ART
DEM	<attr> or <pos>	PDAT
	<subst>	PDS
INDEF	<oD> or <mD>	PIAT
POSS		PPOSAT
PPRO	<pers>	PPER
	<prfl>	PRF
PREP		APPR
PREPART		APPRART
NN		NN
V	<imp>	VAIMP, VMIMP, VVIMP
	<inf> and <zu>	VVIZU
	<inf> and no <zu>	VAINF, VMINF, VVINF
	<ppast>	VAPP, VMPP, VVPP
	no <imp> and no <inf> and no <ppast>	VAFIN, VMFIN, VVFIN
WPRO	<subst>	PWS
	no <subst>	PWAT

Table A.1: Mapping from Morphisto part-of-speech and feature tags to Tiger part-of-speech tags

Appendix B

Annotation of Selector-Target Sets

This appendix describes the algorithm used for the annotation of selector-target sets used in Chapter 7. The algorithm takes a Tiger-style German parse tree as input and extracts a set of selector-target sets. We give the algorithm in procedural form.

Our algorithm uses the constituent labels of the parse trees only; words are ignored. The constituent labels are categorized as shown in Table B.1:

Tree Level	**Category**	**Labels**
Phrase	Subject	*-EP *-SB
Part-of-speech	Comma	,
	FiniteVerb	VAFIN VMFIN VVFIN
	NounModifier	ADJA ART PDAT PIAT PPOSAT PWAT
	NounOrPronoun	NE NN PPER PDS
	ProperNoun	NE
	Preposition	APPR

Table B.1: Definition of constituent categories used in the annotation algorithm.

Figure B.1 shows the top-level function EXTRACT. EXTRACT first calls the sub-procedure EXTRACT-NP-PP to extract the set of selector-target sets that cover noun phrases and prepositional phrases only. EXTRACT-SUBJ-VERB then expands these sets to include finite verbs that are the targets of subject noun phrases. Finally (lines 5 to 9) EXTRACT searchs for nodes that are selector or target types, according to their labels, but have not already been found to belong to a selector-target set. Even though these nodes do not participate in agreement or government relations in this particular parse tree it is important that they are annotated so that constraint induction will include

them in rules extracted from subgraphs of the alignment graph.

```
EXTRACT(root)
1  for each child node n in root's children
2      EXTRACT-NP-PP(n)
3  for each child node n in root's children
4      EXTRACT-SUBJ-VERB(n)
5  for each leaf node n in root's leaves
6      if n already belongs to a selector-target set
7          continue
8      else if ISSELECTORORTARGET(n)
9          add n and PARENT(n) to a new selector-target set

EXTRACT-NP-PP(n)
1  if n does not already belong to a selector-target set
2      if n is a NounPhrase or PrepositionalPhrase
3          Create a new selector-target set, s
4          ADD-NP-PP(n, s)
5  for each child node c in n's children
6      EXTRACT-NP-PP(n)

EXTRACT-SUBJ-VERB(n)
1  if ISCLAUSE(n)
2      ADD-SUBJ-VERB(n)
3  for each child node c in n's children
4      EXTRACT-SUBJ-VERB(c)
```

Figure B.1: Algorithm for extracting selector-target sets (continued in Figure B.2).

The sub-procedures are shown in Figure B.2. ADD-NP-PP first scans the node n's children looking for a head node (line 2). It then scans n's children left-to-right adding modifiers to the selector-target set s. If there are any prepositions then the last preposition (only) is added to the selector-target set.

ADD-SUBJ-VERB scans the node n's children looking for a subject node and a finite verb node. If both are found then it searches for an existing selector-target set containing the subject node. If no such set is found then a new set is created. Finally, the subject, verb, and their parent node are added to the selector-target set.

Finally, Figure B.2 shows the subprocedure FIND-NOUN-PHRASE-HEAD. It scans a node's children looking for a noun or pronoun. If there is a sequence of consecutive nouns or pronouns then the last is taken to be the head.

```
ADD-NP-PP(n, s)
 1  Add n to s
 2  head = FIND-NOUN-PHRASE-HEAD(n)
 3  prep = NIL
 4  for each child node c in n's children
 5      if ISPREPOSITION(c)
 6          prep = c
 7      else if ISNOUNMODIFIER(c)
 8          if prep
 9              add prep to s
10              prep = NIL
11          add c to s
12      else if c == head
13          if prep
14              add prep to s
15          if not ISPROPERNOUN(c)
16              add c to s
17          break
```

```
ADD-SUBJ-VERB(n)
 1  subj = NIL
 2  verb = NIL
 3  for each child node c in n's children
 4      if ISSUBJECT(c)
 5          subj = c
 6      else if ISFINITEVERB(c)
 7          verb = c
 8  if subj and verb
 9      search for a selector-target s containing subj
10      if not found
11          s = new selector-target set
12      add subj to s
13      add verb to s
14      add n to s
```

Figure B.2: Algorithm for extracting selector-target sets (continued from Figure B.1 and continued in Figure B.3).

```
FIND-NOUN-PHRASE-HEAD(n)
 1  head = null
 2  for each child node c in n's children
 3      if head
 4          if ISNOUNORPRONOUN(c)
 5              head = c
 6          else
 7              return head
 8      else if ISNOUNORPRONOUN(c)
 9          head = c
10      else if ISCOMMA(c)
11          break
```

Figure B.3: Algorithm for extracting selector-target sets (continued from Figure B.2).

Appendix C

Annotation of Verbal Complex Sets

This appendix describes the algorithm used for the annotation of verbal complex sets used in Chapter 8. The algorithm takes a Tiger-style German parse tree as input and extracts a set of verbal complex sets. As for the selector-target set extraction algorithm in Appendix B, we give the algorithm in procedural form.

Our algorithm uses the constituent labels of the parse trees only; words are ignored. The constituent labels are categorized as shown in Table C.1:

Tree Level	**Category**	**Labels**
Phrase	Clause	S-* CS-*
	VerbPhrase	VP-OC CVP-OC
Part-of-speech	FiniteVerb	VAFIN VMFIN VVFIN
	Infinitive	VAINF VMINF VVINF
	PastParticiple	VAPP VMPP VVPP
	ClauseEndingPunc	, .

Table C.1: Definition of constituent categories used in the annotation algorithm.

Figure C.1 shows the top-level function EXTRACT. It first calls sub-procedures that make two separate passes over the tree. EXTRACT-PASS-1 checks each clause node for a canonincal verbal complex (as defined by the EXTRACT-FROM-CLAUSE sub-procedure) and creates a verbal-complex set for each one that is found. This process omits some verb phrases, which are mopped up by the second pass in EXTRACT-PASS-2. Any verbs that are still unclaimed are added to their own verbal-complex sets.

The EXTRACT-FROM-CLAUSE and EXTRACT-FROM-VERB-PHRASE sub-procedures are shown in Figure C.2. EXTRACT-FROM-CLAUSE scans a clause node's children

```
EXTRACT(root)
1  EXTRACT-PASS-1(root)
2  EXTRACT-PASS-2(root)
3  for each leaf node n in root's leaves
4      if n already belongs to a verbal complex set
5          continue
6      else if ISVERB(n)
7          add n to a new verbal-complex set

EXTRACT-PASS-1(n)
1  if ISCLAUSE(n)
2      EXTRACT-FROM-CLAUSE(n)
3  for each child node c in n's children
4      EXTRACT-PASS-1(c)

EXTRACT-PASS-2(n)
1  if ISVERBPHRASE(n) and n does not already belong to a set
2      Create new verbal complex set s
3      EXTRACT-FROM-VERB-PHRASE(n,s)
4  for each child node c in n's children
5      EXTRACT-PASS-2(c)
```

Figure C.1: Algorithm for extracting verbal complex sets (continued in Figure C.2).

from left-to-right searching for verbs and verb phrases. If no finite verb is found then the procedure exits early (line 17) and no verbal complex is extracted. Otherwise, an empty verbal complex set is created (line 18) and all verbs and relevant verb phrase nodes are added (lines 19-28). The algorithm assumes that the first child verb phrase containing a finite verb belong to a verbal complex and the others do not (lines 22-24).

EXTRACT-FROM-VERB-PHRASE gathers all verbs and sub-verb phrases from a verb phrase node (line 1) and then adds them to a new verbal complex set (lines 4-6), provided that at least one verb is found.

Finally, Figure C.3 shows the READ-VERBS-FROM-VERB-PHRASE procedure. This scans a verb phrase node's children from left-to-right searching for verbs and nested verb phrases. Normally a verb phrase node will not contain a finite verb and if one is found then the procedure exits early (line 5). The procedure then recursively adds verbs and verb-phrase nodes from nested verb phrases (lines 13-19).

```
EXTRACT-FROM-CLAUSE(n)
 1  finite-verb-list = empty list
 2  non-finite-verb-list = empty list
 3  verb-phrase-list = empty list
 4  for each child node c in n's children, visited left-to-right
 5      if ISFINITEVERB(c)
 6          append c to finite-verb-list
 7      else if ISINFINITIVE(c) or ISPASTPARTICIPLE(c)
 8          append c to non-finite-verb-list
 9      else if ISVERBPHRASE(c)
10          append c to verb-phrase-list
11      else if ISCLAUSEENDINGPUNC(c)
12          if finite-verb-list is empty
13              clear all lists
14          else
15              break
16  if finite-verb-list is empty
17      return
18  create new verbal complex set s
19  add n to s
20  add finite-verb-list[0] to s
21  add all verbs from non-finite-verb-list to s
22  for each vp in verb-phrase-list
23      list-1, list-2 = READ-VERBS-FROM-VERB-PHRASE(vp)
24      if list-1 is not empty
25          Add vp to s
26          Add all verbs from list-1 to s
27          Add all sub-phrase nodes from list-2 to s
28          break

EXTRACT-FROM-VERB-PHRASE(n, s)
1  verbs, subphrases = READ-VERBS-FROM-VERB-PHRASE(n)
2  if verbs is empty
3      return
4  add n to s
5  add all verbs from verbs to s
6  add all sub-phrase nodes from subphrases to s
```

Figure C.2: Algorithm for extracting verbal complex sets (continued from Figure C.2 and continued in Figure C.3).

```
READ-VERBS-FROM-VERB-PHRASE(n)
 1  verb-list = empty list
 2  vp-list = empty list
 3  for each child node c in n's children, visited left-to-right
 4      if ISFINITEVERB(c)
 5          return pair of empty lists
 6      else if ISINFINITIVE(c) or ISPASTPARTICIPLE(c)
 7          append c to verb-list
 8      else if ISVERBPHRASE(c)
 9          append c to vp-list
10      else if ISCLAUSEENDINGPUNC(c)
11          break
12  sub-phrase-list = empty list
13  for each verb phrase vp in vp-list
14      list-1, list-2 = READ-VERBS-FROM-VERB-PHRASE(n)
15      add verbs from list-1 to verb-list
16      add sub-phrases from list-2 to sub-phrase-list
17      if list-1 is not empty
18          add vp to sub-phrase-list
19          break
20  return verb-list, sub-phrase-list
```

Figure C.3: Algorithm for extracting verbal complex sets (continued from Figure C.2).

Bibliography

A. V. Aho and J. D. Ullman. Syntax directed translations and the pushdown assembler. *J. Comput. Syst. Sci.*, 3(1):37–56, February 1969.

Karunesh Kumar Arora and R. Mahesh K. Sinha. Improving Statistical Machine Translation through co-joining parts of verbal constructs in English-Hindi translation. In *Proceedings of the Sixth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 95–101, Jeju, Republic of Korea, July 2012. Association for Computational Linguistics.

Eleftherios Avramidis and Philipp Koehn. Enriching Morphologically Poor Languages for Statistical Machine Translation. In *In Proceedings of ACL, 2008*, 2008.

Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. A maximum entropy approach to natural language processing. *Comput. Linguist.*, 22(1):39–71, 1996.

Ann Bies, Mark Ferguson, Karen Katz, Robert MacIntyre, Victoria Tredinnick, Grace Kim, Mary Ann Marcinkiewicz, and Britta Schasberger. Bracketing Guidelines for Treebank II Style Penn Treebank Project. Technical report, University of Pennsylvania, 1995.

Alexandra Birch, Phil Blunsom, and Miles Osborne. A Quantitative Analysis of Reordering Phenomena. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 197–205, Athens, Greece, March 2009. Association for Computational Linguistics.

E. Black, S. Abney, F. Flickenger, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. A Procedure for Quantitatively Comparing the Syntactic Coverage of English Grammars. In *Proceedings of the Fourth DARPA Speech and Natural Language Workshop*, 1991.

Ondřej Bojar. English-to-Czech factored machine translation. In *StatMT '07: Proceedings of the Second Workshop on Statistical Machine Translation*, pages 232–239, Morristown, NJ, USA, 2007. Association for Computational Linguistics.

Ondřej Bojar and Jan Hajič. Phrase-based and deep syntactic English-to-Czech statistical machine translation. In *StatMT '08: Proceedings of the Third Workshop on Statistical Machine Translation*, pages 143–146, Morristown, NJ, USA, 2008. Association for Computational Linguistics.

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. The TIGER treebank. In *Proceedings of the workshop on treebanks and linguistic theories*, pages 24–41, 2002.

Joan Bresnan. *The Mental representation of grammatical relations*. MIT press series on cognitive theory and mental representation. MIT press, Cambridge (Mass.), London, 1982.

Joan Bresnan. *Lexical-Functional Syntax*. Blackwell Publishers, Oxford, UK, 2001.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. A statistical approach to machine translation. *Comput. Linguist.*, 16(2):79–85, June 1990.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: parameter estimation. *Comput. Linguist.*, 19(2):263–311, June 1993.

David Burkett and Dan Klein. Transforming Trees to Improve Syntactic Convergence. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 863–872, Jeju Island, Korea, July 2012. Association for Computational Linguistics.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. Re-evaluating the Role of BLEU in Machine Translation Research. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational LinguisticsProceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, 2006.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. Findings of the 2012 Workshop on Statistical Machine Translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 10–51, Montréal, Canada, June 2012. Association for Computational Linguistics.

Jean Carletta. Assessing Agreement on Classification Tasks: The Kappa Statistic. *Comput. Linguist.*, 22(2):249–254, June 1996.

Marine Carpuat and Dekai Wu. Improving Statistical Machine Translation Using Word Sense Disambiguation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 61–72, 2007.

J.-C. Chappelier and M. Rajman. A Generalized CYK Algorithm for Parsing Stochastic CFG. In *Proceedings of the First Workshop on Tabulation in Parsing and Deduction*, pages 133–137, 1998.

Stanley F. Chen and Joshua Goodman. An Empirical Study of Smoothing Techniques for Language Modeling. Technical report, Harvard University, 1998.

David Chiang. A hierarchical phrase-based model for statistical machine translation. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 263–270, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

David Chiang. Hierarchical Phrase-Based Translation. *Comput. Linguist.*, 33(2):201–228, 2007.

David Chiang. Learning to Translate with Source and Target Syntax. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1443–1452, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

David Chiang, Kevin Knight, and Wei Wang. 11,001 new features for statistical machine translation. In *NAACL '09: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 218–226, Morristown, NJ, USA, 2009. Association for Computational Linguistics.

Tagyoung Chung, Licheng Fang, and Daniel Gildea. Issues Concerning Decoding with Synchronous Context-free Grammar. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 413–417, Portland, Oregon, USA, June 2011a. Association for Computational Linguistics.

Tagyoung Chung, Licheng Fang, and Daniel Gildea. SCFG Latent Annotation for Machine Translation. In *IWSLT '11*, 2011b.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. Better Hypothesis Testing for Statistical Machine Translation: Controlling for Optimizer Instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 176–181, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

Ann Clifton and Anoop Sarkar. Combining Morpheme-based Machine Translation with Post-processing Morpheme Prediction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Techologies*, pages 32–42, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

J. Cohen. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1):37, 1960.

Michael Collins, Lance Ramshaw, Jan Hajič, and Christoph Tillmann. A Statistical Parser for Czech. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL '99, pages 505–512, Stroudsburg, PA, USA, 1999. Association for Computational Linguistics.

Greville G. Corbett. *Agreement*. Cambridge Textbooks in Linguistics. Cambridge University Press, 2006.

Greville G. Corbett. *Features*. Cambridge Textbooks in Linguistics. Cambridge University Press, 2012.

J. N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. In *The Annals of Mathematical Statistics*, volume 43, pages 1470–1480, 1972.

Steve DeNeefe, Kevin Knight, Wei Wang, and Daniel Marcu. What Can Syntax-Based MT Learn from Phrase-Based MT? In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007). June 28-30, 2007. Prague, Czech Republic.*, 2007.

John DeNero, Mohit Bansal, Adam Pauls, and Dan Klein. Efficient parsing for transducer grammars. In *NAACL '09: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 227–235, Morristown, NJ, USA, 2009. Association for Computational Linguistics.

Jay Earley. An efficient context-free parsing algorithm. *Commun. ACM*, 13(2):94–102, February 1970.

Jason Eisner. Learning Non-Isomorphic Tree Mappings for Machine Translation. In *The Companion Volume to the Proceedings of 41st Annual Meeting of the Association for Computational Linguistics*, pages 205–208, Sapporo, Japan, July 2003. Association for Computational Linguistics.

Mireia Farrús, Marta R. Costa-jussà, José B. Mariño, and José A. R. Fonollosa. Linguistic-based Evaluation Criteria to identify Statistical Machine Translation Errors. In *Proceedings of the 14th Annual Conference of the Euoropean Association for Machine Translation (EAMT'10)*, pages 167–173, May 2010.

Victoria Fossum, Kevin Knight, and Steven Abney. Using syntax to improve word alignment precision for syntax-based machine translation. In *StatMT '08: Proceedings of the Third Workshop on Statistical Machine Translation*, pages 44–52, Morristown, NJ, USA, 2008. Association for Computational Linguistics.

Heidi J. Fox. Phrasal cohesion and statistical machine translation. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10*, EMNLP '02, pages 304–3111, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

Nissim Francez and Shuly Wintner. *Unification Grammars*. Cambridge University Press, New York, NY, USA, 2011.

Alexander Fraser, Marion Weller, Aoife Cahill, and Fabienne Cap. Modeling Inflection and Word-Formation in SMT. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 664–674, Avignon, France, April 2012. Association for Computational Linguistics.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. What's in a Translation Rule? In *HLT-NAACL '04*, 2004.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. Scalable inference and training of context-rich syntactic translation models. In *ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 961–968, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

Qin Gao and Stephan Vogel. Parallel implementations of word alignment tool. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, SETQA-NLP '08, pages 49–57, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

Anita Gojun and Alexander Fraser. Determining the placement of German verbs in English–to–German SMT. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 726–735, Avignon, France, April 2012. Association for Computational Linguistics.

Sharon Goldwater and David McClosky. Improving statistical MT through morphological analysis. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 676–683, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

Yvette Graham, Anton Bryl, and Josef van Genabith. F-structure Transfer-Based Statistical Machine Translation. In *In Proceedings of Lexical Functional Grammar Conference 2009*, 2009.

Spence Green and John DeNero. A Class-Based Agreement Model for Generating Accurately Inflected Translations. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–155, Jeju Island, Korea, July 2012. Association for Computational Linguistics.

Eva Hajičová, Petr Sgall, and Daniel Zeman. Issues of projectivity in the prague dependency treebank. In *Prague Bulletin of Mathematical Linguistics*, 2004.

Greg Hanneman and Alon Lavie. Improving Syntax-Augmented Machine Translation by Coarsening the Label Set. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 288–297, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

Greg Hanneman, Vamshi Ambati, Jonathan H. Clark, Alok Parlikar, and Alon Lavie. An improved statistical transfer system for French–English machine translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, StatMT '09, pages 140–144, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

Kenneth Heafield, Philipp Koehn, and Alon Lavie. Grouping Language Model Boundary Words to Speed K-Best Extraction from Hypergraphs. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Atlanta, Georgia, USA, June 2013.

Maria Holmqvist, Sara Stymne, and Lars Ahrenberg. Getting to know Moses: initial experiments on German–English factored translation. In *StatMT '07: Proceedings of the Second Workshop on Statistical Machine Translation*, pages 181–184, Morristown, NJ, USA, 2007. Association for Computational Linguistics.

Mark Hopkins and Greg Langmead. SCFG Decoding Without Binarization. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 646–655, Cambridge, MA, October 2010. Association for Computational Linguistics.

Mark Hopkins and Jonathan May. Tuning as Ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics.

Liang Huang and David Chiang. Forest Rescoring: Faster Decoding with Integrated Language Models. In *ACL*, 2007.

Liang Huang and Haitao Mi. Efficient Incremental Decoding for Tree-to-String Translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 273–283, Cambridge, MA, October 2010. Association for Computational Linguistics.

Liang Huang, Kevin Knight, and Aravind Joshi. A syntax-directed translator with extended domain of locality. In *Proceedings of the Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing*, CHSLP '06, pages 1–8, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

Liang Huang, Hao Zhang, Daniel Gildea, and Kevin Knight. Binarization of synchronous context-free grammars. *Comput. Linguist.*, 35(4):559–595, 2009.

Minwoo Jeong, Kristina Toutanova, Hisami Suzuki, and Chris Quirk. A Discriminative Lexicon Model for Complex Morphology. In *The Ninth Conference of the Association for Machine Translation in the Americas*. Association for Computational Linguistics, November 2010.

R. S. Z. Kalijahi, R. Rubino, J. Roturier, and J. Foster. A Detailed Analysis of Phrase-based and Syntax-based Machine Translation: The Search for Systematic Differences. In *In Proceedings of Tenth Biennial Conference of the Association for Machine Translation in the Americas (AMTA-2012), San Diego, CA.*, 2012.

T. Kasami. An efficient recognition and syntax algorithm for context-free languages. Technical Report AFCLR-65-758, Air Force Cambridge Research Laboratory, Bedford, MA, 1965.

Martin Kay. Functional Grammar. In *Proceedings of the Fourth Annual Meeting of the Berkeley Linguistics Society*, 1979.

Martin Kay. Functional Unification Grammar: A Formalism for Machine Translation. In *Annual Meeting of the Association of Computational Linguistics*, 1984.

Dan Klein and Christopher D. Manning. Parsing and hypergraphs. In *In IWPT*, pages 123–134, 2001.

Kevin Knight. Decoding complexity in word-replacement translation models. *Comput. Linguist.*, 25(4):607–615, December 1999.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. French, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. Open Source Toolkit for Statistical Machine Translation: Factored Translation Models and Confusion Network Decoding. Technical report, Final Report of the 2006 Language Engineering Workshop, Johns Hopkins University., 2007a.

Philipp Koehn. Europarl: A Parallel Corpus for Statistical Machine Translation. In *MT Summit 2005*, 2005.

Philipp Koehn and Hieu Hoang. Factored Translation Models. In *In Proceedings of EMNLP, 2007*, 2007.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 48–54, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

Philipp Koehn, Amittai Axelrod, Alexandra Birch, Chris Callison-Burch, Miles Osborne, and David Talbot. Edinburgh System Description Edinburgh System Description for the 2005 IWSLT Speech Translation Evaluation. In *In Proceedings of IWSLT*, 2005.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Morristown, NJ, USA, 2007b. Association for Computational Linguistics.

Alon Lavie. Stat-XFER: a general search-based syntax-driven framework for machine translation. In *Proceedings of the 9th international conference on Computational linguistics and intelligent text processing*, CICLing'08, pages 362–375, Berlin, Heidelberg, 2008. Springer-Verlag.

Young-Suk Lee. Morphological analysis for statistical machine translation. In *HLT-NAACL '04: Proceedings of HLT-NAACL 2004: Short Papers on XX*, pages 57–60, Morristown, NJ, USA, 2004. Association for Computational Linguistics.

Mike Lewis and Mark Steedman. Combined Distributional and Logical Semantics. *Transactions of the Association for Computational Linguistics*, 1:179–192, May 2013.

Yang Liu, Qun Liu, and Shouxun Lin. Tree-to-string alignment template for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 609–616, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

Daniel Marcu and William Wong. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10*, EMNLP '02, pages 133–139, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

Daniel Marcu, Wei Wang, Abdessamad Echihabi, and Kevin Knight. SPMT: statistical machine translation with syntactified target language phrases. In *EMNLP '06: Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 44–52, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

Yuval Marton and Philip Resnik. Soft Syntactic Constraints for Hierarchical Phrased-Based Translation. In *Proceedings of ACL-08: HLT*, pages 1003–1011, Columbus, Ohio, June 2008. Association for Computational Linguistics.

Haitao Mi and Liang Huang. Forest-based Translation Rule Extraction. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 206–214, Honolulu, Hawaii, October 2008. Association for Computational Linguistics.

Einat Minkov, Kristina Toutanova, and Suzuki Hisami. Generating Complex Morphology for Machine Translation. In *Proceedings of the ACL*, 2007.

Sonja Nießen and Hermann Ney. Toward hierarchical models for statistical machine translation of inflected languages. In *Proceedings of the workshop on Data-driven methods in machine translation*, pages 1–8, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

Joakim Nivre and Jens Nilsson. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 99–106, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

Franz Josef Och. *Statistical Machine Translation: From Single-Word Models to Alignment Templates*. PhD thesis, RWTH Aachen, Germany, 2002.

Franz Josef Och. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 160–167, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 295–302, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Comput. Linguist.*, 29(1):19–51, March 2003.

Franz Josef Och and Hermann Ney. The Alignment Template Approach to Statistical Machine Translation. *Comput. Linguist.*, 30(4):417–449, 2004.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

Fernando C. N. Pereira. A structure-sharing representation for unification-based grammar formalisms. In *Proceedings of the 23rd annual meeting on Association for Computational Linguistics*, pages 137–144, Morristown, NJ, USA, 1985. Association for Computational Linguistics.

Slav Petrov and Dan Klein. Improved Inference for Unlexicalized Parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411, Rochester, New York, April 2007. Association for Computational Linguistics.

Carl Pollard and Ivan A. Sag. *Head-Driven Phrase Structure Grammar*. The University of Chicago Press, Chicago, 1994.

Ananthakrishnan Ramanathan, Hansraj Choudhary, Avishek Ghosh, and Pushpak Bhattacharyya. Case markers and morphology: addressing the crux of the fluency problem in English-Hindi SMT. In *ACL-IJCNLP '09: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2*, pages 800–808, Morristown, NJ, USA, 2009. Association for Computational Linguistics.

Owen Rambow. The Simple Truth about Dependency and Phrase Structure Representations: An Opinion Piece. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 337–340, Los Angeles, California, June 2010. Association for Computational Linguistics.

Jason Riesa, Ann Irvine, and Daniel Marcu. Feature-Rich Language-Independent Syntax-Based Alignment for Statistical Machine Translation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 497–507, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics.

Stefan Riezler and John T. Maxwell, III. Grammatical machine translation. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 248–255, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

Brian Roark, Kirsty Hollingshead, and Nathan Bodenstab. Finite-State Chart Constraints for Reduced Complexity Context-Free Parsing Pipelines. *Computational Linguistics*, 2012.

Libin Shen, Jinxi Xu, and Ralph Weischedel. A New String-to-Dependency Machine Translation Algorithm with a Target Dependency Language Model. In *Proceedings of ACL-08: HLT*, pages 577–585, Columbus, Ohio, June 2008. Association for Computational Linguistics.

Stuart M. Shieber. The design of a computer language for linguistic information. In *Proceedings of the 10th international conference on Computational linguistics*, COLING '84, pages 362–366, Stroudsburg, PA, USA, 1984. Association for Computational Linguistics.

Stuart M. Shieber. *An Introduction to Unification-Based Approaches to Grammar*, volume 4 of *CSLI Lecture Notes Series*. Center for the Study of Language and Information, Stanford, CA, 1986.

Stuart M. Shieber and Yves Schabes. Synchronous tree-adjoining grammars. In *Proceedings of the 13th conference on Computational linguistics - Volume 3*, COLING '90, pages 253–258, Stroudsburg, PA, USA, 1990. Association for Computational Linguistics.

Andreas Stolcke. SRILM - An Extensible Language Modeling Toolkit. In *Intl. Conf. Spoken Language Processing, Denver, Colorado, September 2002.*, 2002.

Sara Stymne, Maria Holmqvist, and Lars Ahrenberg. Effects of morphological analysis in translation between German and English. In *StatMT '08: Proceedings of the Third Workshop on Statistical Machine Translation*, pages 135–138, Morristown, NJ, USA, 2008. Association for Computational Linguistics.

Michael Subotin. An exponential translation model for target language morphology. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Techologies*, pages 230–238, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

David Talbot and Miles Osborne. Modelling lexical redundancy for machine translation. In *ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 969–976, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

Christoph Tillmann. A unigram orientation model for statistical machine translation. In *Proceedings of HLT-NAACL 2004: Short Papers*, HLT-NAACL-Short '04, pages 101–104, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

Kristina Toutanova, Hisami Suzuki, and Achim Ruopp. Applying Morphology Generation Models to Machine Translation. In *Proceedings of ACL, Association for Computational Linguistics, June 2008*, 2008.

David Vilar, Jia Xu, Luis Fernando D'Haro, and Hermann Ney. Error Analysis of Machine Translation Output. In *International Conference on Language Resources and Evaluation*, pages 697–702, Genoa, Italy, may 2006.

Wei Wang, Jonathan May, Kevin Knight, and Daniel Marcu. Re-structuring, relabeling, and re-aligning for syntax-based machine translation. *Comput. Linguist.*, 36(2):247–277, 2010.

Jonathan Weese, Chris Callison-Burch, and Adam Lopez. Using categorial grammar to label translation rules. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, WMT '12, pages 222–231, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

Marion Weller, Alexander Fraser, and Sabine Schulte im Walde. Using subcategorization knowledge to improve case prediction for translation to German. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 593–603, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

Philip Williams. Towards Statistical Machine Translation with Unification Grammars. Master's thesis, University of Edinburgh, UK, 2009.

Philip Williams and Philipp Koehn. Agreement Constraints for Statistical Machine Translation into German. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 217–226, Edinburgh, Scotland, July 2011. Association for Computational Linguistics.

Philip Williams and Philipp Koehn. GHKM Rule Extraction and Scope-3 Parsing in Moses. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 388–394, Montréal, Canada, June 2012. Association for Computational Linguistics.

Philip Williams and Philipp Koehn. Using Feature Structures to Improve Verb Translation in English-to-German Statistical MT. In *Proceedings of the 3rd Workshop on Hybrid Approaches to Machine Translation (HyTra)*, pages 21–29, Gothenburg, Sweden, April 2014. Association for Computational Linguistics.

D Wroblewski. Nondestructive Graph Unification. *In Proceedings of AAAI87*, 1987.

Fei Xia and Martha Palmer. Converting Dependency Structures to Phrase Structures. In *Proceedings of the First International Conference on Human Language Technology Research*, HLT '01, pages 1–5, Stroudsburg, PA, USA, 2001. Association for Computational Linguistics.

Kenji Yamada and Kevin Knight. A syntax-based statistical translation model. In *ACL '01: Proceedings of the 39th Annual Meeting on Association for Computational*

Linguistics, pages 523–530, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

Richard Zens, Franz Josef Och, and Hermann Ney. Phrase-Based Statistical Machine Translation. In *Proceedings of the 25th Annual German Conference on AI: Advances in Artificial Intelligence*, KI '02, pages 18–32, London, UK, 2002. Springer-Verlag.

Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. Synchronous binarization for machine translation. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 256–263, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

Jiajun Zhang, Feifei Zhai, and Chengqing Zong. Augmenting String-to-Tree Translation Models with Fuzzy Use of Source-side Syntax. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 204–215, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics.

Andrea Zielinski and Christian Simon. Morphisto –An Open Source Morphological Analyzer for German. In *Proceeding of the 2009 conference on Finite-State Methods and Natural Language Processing: Post-proceedings of the 7th International Workshop FSMNLP 2008*, pages 224–231, Amsterdam, The Netherlands, The Netherlands, 2009. IOS Press.

Andreas Zollmann and Ashish Venugopal. Syntax augmented machine translation via chart parsing. In *StatMT '06: Proceedings of the Workshop on Statistical Machine Translation*, pages 138–141, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

Andreas Zollmann, Ashish Venugopal, and Stephan Vogel. Bridging the inflection morphology gap for Arabic statistical machine translation. In *NAACL '06: Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers on XX*, pages 201–204, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

Andreas Zollmann, Ashish Venugopal, Franz Och, and Jay Ponte. A systematic comparison of phrase-based, hierarchical and syntax-augmented statistical MT. In *COLING '08: Proceedings of the 22nd International Conference on Computational Linguistics*, pages 1145–1152, Morristown, NJ, USA, 2008. Association for Computational Linguistics.

Arnold Zwicky. German adjective agreement in GPSG. *Linguistics*, 1986.